aal-2014-171

SENIOR-TV

PROVIDING ICT-BASED FORMAL AND INFORMAL CARE AT HOME

Deliverable D-2.3
SENIOR-TV V3

Document information		
Due date of deliverable	30/09/2018	
Actual submission date	06/03/2019	
Organization name of lead contractor for this deliverable	IMATIA	
Revision	15/04/2019	
Dissemination Level		
PU	Public	X
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Authors list	
Author	**Partner**
Iulian Anghelache	CPX
Giorgos Kostopoulos	GULK
Sira López	IMATIA
Yago Martínez	IMATIA
Carlos Rivas	IMATIA/UVigo
Iztok Žilavec	RC-IKTS

Peer Reviewers	
Reviewer	**Partner**
Luis Anido	Advisory Board Member
Cibran Ledo	IMATIA

Versioning	
Version	**Summary**
V_1.0	SENIOR-TV platform from M1 to M35
V_2.0	Include Cibrán Ledo suggestions
V_3.0	Include Luis Anido suggestions

This project has been funded with support from the European Commission.

This document reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

0. Executive summary

The main findings obtained during SENIOR-TV platform development are the following:

1. Involving the users during development process is essential in order to get a successful business product. Not only the final user (seniors) but also the secondary users (relatives, caregivers, etc.) because every stakeholder has their own expectations and needs to incorporate to the final product.
2. Using a progressive development allows to detect early design errors that could generate a failed product. The services developed during the first pilot were oriented to the remote control with pointer, which finally proved very difficult to hand by the seniors. Once the problem was detected, during the next iterations, all the interfaces' navigation was simplified and reduced in order to use only the OK, BACK and D-pad keys. The use of pointer in the final product would have generated a non-viable product for the elderly.
3. People over 65 are not a homogeneous group, they have different concerns and needs. Therefore, the SENIOR-TV platform has had to include services of diverse nature to cover all different identified aspects of the elderly lives: reduce loneliness and keep in touch with the external world (services such as News, Weather or Video chat), keep socially and physically active (promoting the participation of local activities), maintain an active mind through entertainment games and, encourage a healthy life (controlling health basic parameters or doing exercise using Tracker or Virtual Center services).
4. The interface TV screens must be as simple as possible. The information shown must be easy and concise reducing the use of long texts. In addition, the number of actions to be carried out on each screen must be limited and its purpose have to be clearly defined.
5. Maintaining the service contents updated is basic to continue using the platform and not to lose the elderly interest. Contents must be localized considering the country where the pilots are going to be performed because each country has its own particularities and the society can have different interests and concerns. However, the data security is relevant independently of country.
6. Seniors really appreciate that services are translated to their local languages. App translation is basic to use it properly even when it is only a pilot that is made. Translate the services names helps to understand their purpose.
7. Using touch devices instead of traditional remote control to manage the TV apps, can contribute to reduce the SENIOR-TV rejection, facilitating the interaction with the platform and making easy the access of its contents.
8. The services development in parallel, allowed to advance quickly in the first versions of the platform, and to obtain the feedback of the majors from the first phases of the development.

However, seniors did not perceive SENIOR-TV as a platform, but as a set of decoupled services. Finally, this point has tried to be solved with the development of the Client Ecosystem and its adhesion with the System Integration.

9. Considering the current smart-TV market fragmentation, creating a platform widely compatible with different operating systems is a big challenge. Even more, considering the SENIOR-TV low-cost restriction. Focusing on the Android TV platform and hybrid technologies allowed to reduce development time and to invest more time to adapt the TV interaction and interface to the seniors' particular needs. At the same time, hybrid technologies allow to adapt the developments to another non-Android system, if would be necessary.

1. Introduction

This document summarizes the work performed from month M1 to M35 in WP2, which result is the final version of SENIOR-TV platform. This deliverable incorporates the description of done work in WP2 during months M27 to M35 to the work done in the previous deliverables D2.1 and D2.2. In particular, the developments resulted by the third platform version are include in sections 5.3 SENIOR-TV V3 (Informal services), 6.3. SENIOR-TV V3 (Formal services), 7.3.1 Touch Remote Control, 8.1.4. User, roles and relations API and 8.2. Client TV Ecosystem.

The main objective of WP2 is to produce a platform based on interactive TV and mobile technologies according to the requirements of openness and low-cost. The obtained platform has to cover the general project aims: providing formal and informal caregiving services for older adults, focusing on the active prevention and the maintenance of relationships with their friends, their relatives and, the community. These are the tasks developed in the WP2:

- Task 2.1: Smart-TV Application Framework (M1-33).
- Task 2.2: Design and development of informal services (M3-M34).
- Task 2.3: Design and development of formal services (M3-M34).
- Task 2.4: Ergonomic design (M4-M33).
- Task 2.5: System Integration (M4-M35).

The following sections include a detailed description of performed work in each of the tasks. Sections 2 and 3 include the details of technology analysis. Sections 4, 5, 6, 7 and 8 present the developed services, the system architecture and the graphical interface design. Finally, the conclusions and annexes are included on sections 9 and 10.

2. Technology analysis

During first annuity, a study of TV platforms and development technologies was done. As a result of this research, the Android operating system and a Minix STB[1] were selected to build the SENIOR-TV V1. In addition, the first services were developed using web hybrid technologies.

Considering those decisions, the SENIOR-TV V2 and V3 services have developed using Android native and hybrid web technologies. In addition, Android native technology was used to create the TV Client Ecosystem during the third annuity.

2.1. Smart-TV software

In this section, we introduce the main environments for application development in the smart-TV world resulting from first annuity study. While early approaches were based on proprietary systems, present-day, more elaborated solutions are based on Linux, and in some cases, follow the-web-as-a-platform paradigm based on technologies such as HTML5, CSS or JavaScript. The most relevant proposals available at the time of this writing are introduced below: AndroidTV, WebOS, FireOS, Firefox OS, Tizen. A final summary is also included to compare the characteristics of the approaches discussed.

1. **Android**[2] is a Linux-based operating system developed by the Open Handset Alliance, a consortium of developers led by Google. In June 2014, Google introduced AndroidTV[3] as an evolution of Android version 5.0 Lollipop to be deployed in television environments. Currently, manufacturers like Sony, Sharp and Philips have shown interest in including this operating system in their appliances, being also possible to acquire distributions in USB sticks to be connected directly to a media input (e.g., HDMI).

   Application programming for smart televisions in Android follows the same model as smartphone programming. It is based on the Java programming language and supports user interfaces definition through XML files. Frameworks are also available to support the development of web applications based on HTML5, although these applications do not have unlimited access to operating system services as native applications do. AndroidTV developers must explicitly indicate that their applications are designed for a TV environment. Typical features of a mobile phone that are not available on a TV set (e.g., touch screen, GPS receiver, camera, etc.) may be removed from the interface requirements in the application manifest to facilitate application migration to a TV

[1] http://minix.com.hk/
[2] Android official site: https://www.android.com/
[3] AndroidTV official site: https://www.android.com/tv/

environment. AndroidTV users will access the Google application repository (i.e., Google play) to download new applications to their TV sets in a similar way to Android smartphones.

2. **WebOS**[4] is an operating system developed by Palm also based on the Linux operating system. This platform was acquired by Hewlett-Packard to be integrated into its devices. With the advent of Android, Hewlett-Packard halted WebOS development and eventually sold it to LG, which included it into its smart-TV sets.

   Thus, WebOS is an operating system designed for integration in smartphones and other devices such as smart-TVs. Application development is based on web standards like HTML5, JavaScript and CSS. Its design allows web applications to access the native functionality of the host system. Applications are executed through the webkit renderer, and the system is a multitasking one able to run up to 4 simultaneous applications. As a general rule, a WebOS application is composed of HTML5 code and functional blocks written in JavaScript. While there is an open source distribution named Open WebOS, the version integrated in LG appliances is closed. For application development in this environment, LG offers the Enyo Framework to the developer's community.

   TV sets based on WebOS include a remote-control equipped with a motion sensor that sets a pointer on the television screen in the style of the Nintendo Wii. Additionally, it implements a voice recognition system that allows voice interaction using certain commands. Users access the LG store to download applications.

3. **FireOS**[5] is a Linux-based operating system developed by Amazon for mobile devices (e.g., Kindle) and smart-TV devices like FireTV. It is a version of the Android operating system, specifically the Fire OS 3.0 version based on Android Jelly Bean (API level 17). It can be installed in Amazon devices only.

   Applications are written in Java and can be downloaded from the Amazon app store. From the point of view of the application developer, this platform is analogous to Android TV, with the particularity that the applications are available exclusively from the Amazon store. While the operating system is based on an Android distribution, Amazon limits its installation to its own hardware. This, together with the difficulty of downloading apps from other repositories, dramatically reduces the target audience of this platform.

4. **Firefox** OS[6] is an open source operating system based on Linux and the Mozilla Gecko technology. While initially designed for mobile devices, some smart-TV manufacturers like Panasonic have

[4] WebOS official site: http://www.lg.com/uk/smarttv/index.html
[5] FireOS official site: https://developer.amazon.com/android-fireos
[6] Firefox OS official site: https://www.mozilla.org/en-US/firefox/os/

already shown their interest in this platform. At the time of booting, the Gecko runtime environment is launched, thus allowing the execution of web applications. Developers may use HTML5, CSS and JavaScript standards for their developments. Each application installed on this operating system behaves like a web page. Firefox OS also introduced the WebAPI concept, through which a selection of system calls can be invoked from the web interface. This reduces the gap between current web applications and native applications that can be installed on this operating system.

5. **Tizen**[7] is a Linux-based open source operating system supported by the Linux Foundation and companies such as Intel, Samsung, Huawei and Fujitsu, among others. While Tizen was initially conceived as a platform for developing web-based applications, from version 2.0 on it also supports native applications. With this enhancement, support for hybrid applications was also included. These latter applications include native components and web application components. Native application development is done in C++. For web applications, HTML5, CSS and JavaScript technologies may be used. Through a WebAPI web applications may access certain core features (e.g., Bluetooth support). The execution environment is based on webkit rendering. Samsung provides an SDK for application design. The ability to run native applications offers many advantages when it comes to access and interact with both the operating system and with other devices that can be connected to the TV set. Similarly, support of standards-based web applications greatly facilitates application design and deployment. The blend of these two approaches in hybrid applications allows programmers to take advantage of both paradigms.
6. **Google TV** created high expectations due to the relevance of the developing company (Google Inc.). However, Google abandoned this line in favour of Android TV. In our case, a Google TV device was evaluated and discarded due to the fact that interactivity is achieved through web browsers, which in turn poses significant restrictions on the access to operating system resources.
7. **Apple TV** is the TV software created by Apple Inc. In the case of Apple TV, application distribution is based on the Apple Store concept. Convenient support is already available to develop applications for it, but there are also very significant restrictions when connecting external devices. Also, in accordance with the user requirements, in order to provide them with apps with a great interaction experience and a rich set of features, this technology was disregarded.

Android is the base operating system for AndroidTV and FireOS. As a consequence, they can benefit from a rich set of development applications and an active developer's community (in the case of FireOS applications are limited to those available through the Amazon App Store).

[7] Tizen official site: https://www.tizen.org/

Another identified trend is the adoption of web-as-a-platform paradigm by WebOS, FirefoxOS and Tizen. As in the previous case, many tools are readily available to support development. This, together with the Web APIs offered by these systems to provide access to the internals of the operating system, supports the development of complete and rich interactive applications.

All discussed platforms, either Android-based or web-as-a-platform, offer development environments similar to those found in the smartphone world. Thus, support is provided to access peripheral devices that will enhance the user experience (e.g., webcams, voice control, gesture control, etc.). In some cases, the integration of external applications with direct access to native functionalities is also possible to access devices like webcams or microphones.

In the smart-TV world, there are application areas that require a more direct control of the operating system and the TV itself. For example, interaction with some applications has to take place while watching a broadcast channel. While this is not a dramatic limitation in the smartphone world, it is a relevant feature in the case of interactive TV (e.g. contests/games applications based on a TV show).

2.2. Smart-TV devices

Smart-TV is a special type of television with an operating system which adds to the TV a set of typical computer characteristics. Most of the smart-TVs on the today's market have one of the operating systems reviewed in the previous section.

During the last years, each television manufacturer has bet for a particular operating system, even has supported its development economically. Therefore, there is a clear relationship between the TV brand and its software. Next table shows the environments of the main European television brands:

TV Brand	Operating system
LG	webOS
Panasonic	Firefox OS
SAMSUNG	TIZEN
SONY	androidtv

The main risk of buying a smart-TV is the market volatility, because it is evolving continuously and there is not a clearly dominant technology or brand. Android is the operating system that presents more facilities to develop services for it. However, the best-selling television brands in Europe are Samsung and LG[8] which have their own software - Tizen and WebOS respectively.

Apart from smart-TVs, there are other devices on the market that allow you to convert any television with an HDMI port into a smart-TV. Most of these devices have an Android operating system, but there are devices with Apple-TV or even Tizen. In general, these devices are cheaper than smart-TVs, therefore they are a great alternative to enjoy the advantages of smart-TVs at a low cost.

Here are different software and hardware parameters to consider when selecting the device to covers the formal and informal service requirements. The formal services are the most demanding because they use special sensors to communicate with devices. The main important issues to review are:

- Operating systems.
- Openness.
- Easiness develops.
- Supports modify the start.
- In/out ports (Bluetooth, HDMI, Wi-Fi USB, etc.).
- Internal and SD storage.
- Memory size and processing capacity.
- External peripherals support (webcam, microphone, remote-control, etc.).
- Tv antenna.

The tables below present the general characteristics of the HDMI devices analysed during months M1 to M6.

Hardware characteristics										
Device name	**Memory GB**	**Storage GB**	**Wi-Fi**	**BT**	**HDMI**	**SD**	**USB**	**Webcam**	**External peripherals**	**RF input**
Chromebit CS10	2	16	yes	yes	yes	no	yes	yes	yes	no
Rikomagic v5	2	16	yes	yes	yes	yes	yes	yes	yes	no
Guleek A8	2	8	yes	no	yes	yes	yes	no	yes	no
Nvidia Shield	3	500	yes	yes	yes	yes	yes	yes	yes	no
Amazon Fire TV	2	8	yes	yes	yes	yes	yes	no	yes	no

[8] https://www.statista.com/statistics/267095/global-market-share-of-lcd-tv-manufacturers/

Samsung HomeSync		1 TB	yes	yes	yes	no	yes	no	no	no
Apple TV		32/64	yes	yes	yes	no	yes	no	yes	no
Nexus player	1	8	yes	yes	yes	no	yes	no	yes	no
Raspberry pi 3	1		yes	yes	yes	yes	yes	yes	yes	no
Google TV Box MINIX NEO	2	16	yes	yes	yes	yes	yes	yes	yes	no
MINIX Neo X8-H Plus	2	16	yes	yes	yes	yes	yes	yes	yes	no
Roku4	512Mb	0	yes		yes	yes	yes	no	no	no
Asus Cube v2			yes	no	yes	no	yes	no		no
Boxee Box	1		yes	no	yes	yes	yes	no		no
Udoo	1	SD	yes	yes	yes	yes	yes	yes		no

Software characteristics									
Device name	**OS**	**Cordova**	**SDK**	**app store**	**3rd party libs**	**Openness**	**Easy dev**	**Community**	**Modify start**
Chromebit CS10	Chrome OS	yes	yes	yes	no	yes	yes	no	no
Rikomagic v5	Android, Ubuntu	yes	yes	yes	no	yes	yes	no	yes
Guleek A8	Android	yes	yes	yes	no	yes	yes	no	no
Nvidia Shield	Android TV	yes	yes	yes	yes	yes	yes	yes	no
Amazon Fire TV	Fire OS	yes	yes	yes	yes	yes	yes	yes	yes
Samsung HomeSync	Android	yes	yes	yes	no	yes	yes	no	no
Apple TV	tvOS	no	yes	yes	yes	no	no	yes	no
Nexus player	Android	yes	yes	yes	no	yes	yes	yes	yes
Raspberry pi 3	Windows, Linux	yes/no	no	yes	yes	no/yes	yes	yes	yes
Google TV Box MINIX NEO	Android	yes	yes	yes	yes	yes	yes	yes	yes
MINIX Neo X8-H Plus	Android	yes	yes	yes	yes	yes	yes	yes	yes

Roku4	RokuOS	no			yes	no	yes	yes	no
Asus Cube v2	Android TV	yes	yes	yes	no	yes	yes	no	no
Boxee Box	Linux	yes	no	no	yes	yes	yes	yes	yes
Udoo	Android, Linux, Arduino	yes	yes	yes		yes	yes		yes

After the market analysis, WP2 partners have decided to **acquire HDMI devices with** Android operating system. This decision was founded in three main reasons:

- They are low-cost devices whose software can be upgraded. This property reduces the risk of investing a large amount of money in devices that could be obsolete at the end of the project.
- Android supports applications developed using standard web technologies (see section 3) which allows reusing not only the code but also the development experience for other operating systems and devices.
- It is possible to modify the Android boot to run apps when devices start.

3. Development technologies

This section includes a general overview of the different technologies used to develop the SENIOR-TV platform. Not only to implement the formal and informal services but also the backend cloud and TV ecosystem.

3.1. Smart-TV development technologies

The smart-TV market is changing every day. As we have seen in section 2, there is a great variety of smart-TV technologies which are progressing continuously. Therefore, there is not a unique approach to develop apps for Smart-TV and, the current approaches are based in to reused or extend the mobile app development. This is a valid approach because, as we have seen, the Smart-TV operating systems are almost the same of smartphone software: Android, iOS, Tizen, etc. with some special characteristic oriented to run in televisions.

There are three typical approaches to implement mobile apps and all of them can be used to develop smart-TV apps:

1. **Native apps**: are developed in a programming language native to the device and operating system and require one specific app to be created for one target platform.
2. **Cross-platform apps**: use an intermediate language that is not native to the device's operating system, such as JavaScript, to implement the apps. This allows share some code across different platforms – for instance, across iOS and Android.

3. **Hybrid web apps**: are cross-platform apps which rendering the user interface using an embedded web browser and, which are developed using well-known technologies - HTML, CSS and JavaScript.

The choice of one of these options depending on different aspects such us the performance and hardware requirements, supporting devices and operating systems, available resources or economic restrictions.

1. **Native** apps are appropriate if you need a high-level performance and speed, for instance graphics intensive apps like games. Or to use advanced features offered by the device and operating system (data storage, memory access or complex networking).

   However, this approach is not indicated to support multiple devices, because needs to create different versions for each operating system. So, it requires a bigger investment for development, maintenance and testing.
2. **Cross-platform** is indicated for create apps which must run in different platforms and have high interface performance restrictions. Reduces the development time because the code can be shared between different versions of the apps across devices. However, delegates the access to the device and operating system features to the framework, so, in the end, the performance could be reduced.
3. **HTML5 Hybrid** is the best choice when apps have to run on different devices and operating systems. The base of this approach is the use of basic web technologies such as HTML, CSS and JavaScript, but you can use more advanced libraries such as AngularJS or Bootstrap. Therefore, this paradigm allows us to create advanced interfaces with little effort. In addition, you can reuse most of the code, therefore the development time is greatly reduced.

   However, as in the case of Cross-platform, the access to the device and operating system is delegating to the framework, which reduces the response time. So, it is not recommended for implementing services which need a high level of performance.

The great variety of devices and operating systems presents in smart-TV market makes necessary using technologies which cover as many brands as possible. Neither HTML5 Hybrid nor Cross-platform approaches cover all the operating systems in the market. However, all the TV apps of informal services have been developed with any of these technologies: Unity - a cross-platform technology - to implement the *Attentix* and *Episodix* games and, JavaScript, Angular or Ionic with Apache Cordova for the rest of services.

On the one hand, the particular needs of formal services (integration with external devices) and the Client Ecosystem (launching the TV apps) made necessary the use of native technologies to implement them.

The following subsections show the most relevant characteristics of the selected technologies to developed the SENIOR-TV services.

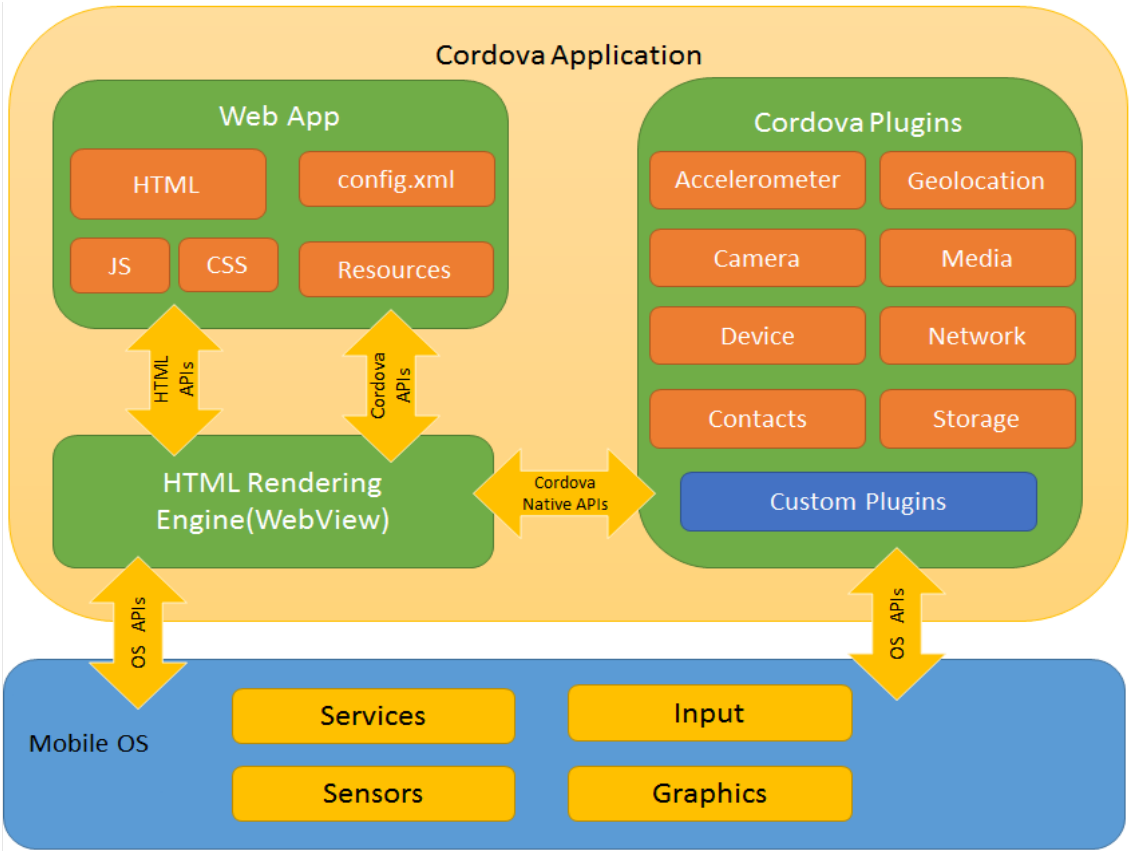
3.1.1 Apache Cordova

Apache Cordova[9] is an open-source mobile development framework created by Nitobi (now Adobe Inc.[10]). Originally called PhoneGap it was bought by Adobe Systems in 2011, and later released as an open source version called Cordova. Contributors to the Apache Cordova project include, among others, Mozilla, BlackBerry, Google, IBM, Intel and, Microsoft.

Apache Cordova allows software programmers to use standard web technologies (HTML5, CSS3, and JavaScript) for cross-platform development. Instead of relying on platform-specific APIs like those in Android, iOS, etc. The resulting applications are hybrid because use webviews to render the layouts and they are packaged as apps for distribution and have access to native device APIs.

Applications execute within wrappers targeted to each platform, and rely on standards-compliant API bindings to access each device's capabilities such as accelerometer, camera, compass, file system, microphone, network status, etc.

The following diagram shows a high-level view of an Apache Cordova application architecture:



[9] Apache Cordova official site: https://cordova.apache.org/
[10] https://www.adobe.com/

FIGURE 1. APACHE CORDOVA ARCHITECTURE

These are the Cordova architecture main components:

- Web App: the application code. Uses CSS3 and HTML5 to create the user interfaces and JavaScript implement the application logic.
- WebView: minimal browser that delivers web content and rendering the apps. HTML5 provides access to underlying hardware such as sensors, data or GPS. However, browsers' support is not consistent across mobile browsers of the oldest versions of operating systems. To overcome these limitations, Apache Cordova embeds the HTML5 code inside a native webview on the device.
- Plugins: provide an interface for Cordova and native components to communicate with each other, allowing invoke native code from JavaScript. Apache Cordova allows developers to extend its functionalities using native plugins that can be called from JavaScript[11].

FIGURE 2. APACHE CORDOVA PLUGINS

The supported platforms by Cordova have been changing over the last few years. Previous versions have supported Android, iOS, Samsung Tizen, WebOS, Firefox OS, etc. However, only Android, Blackberry, iOS, OS X, Ubuntu and Windows 8 and 10 are supported in the last version. These limitations are not a deep problem, because Firefox OS is now discontinued. Also, Tizen has similar Cordova tools to develop hybrid apps. So, we can reuse the HTML5, CSS and most of JavaScript code to create Tizen apps.

However, the new Apple tvOS doesn't use webviews, so it is not possible develop apps using web resources, only native code is allowed. Therefore, this could be a potential problem to analyse in future versions of SENIOR-TV platform.

[11] Cordova Plugin development Guide: https://cordova.apache.org/docs/en/latest/guide/hybrid/plugins/

3.1.2. Ionic Framework

Ionic[12] is a powerful HTML5 SDK that is focused mainly on the look and feel, and UI app interaction. It helps you build native-feeling mobile apps using web technologies like HTML, CSS, and JavaScript. It doesn't replace Cordova but fits in well with Cordova projects in order to simplify one big part of the app: the front-end.

Ionic was created by Drifty Co. in 2013. As Ionic is based on Angular, currently exists two versions of the framework in parallel: a stable version upon first generation of AngularJS (first release released in May 2015) and, a beta version based on Angular 2 (a new version that has not been released as stable until a few months ago).

Ionic includes mobile components, typography, interactive paradigms, and an extensible base theme. It also provides Angular custom components such as collection repeat, scroll-view, etc. Users can build their apps and customize for their favourite operating system and then deploy the apps using Cordova.

The last version supports several systems like Android 4.1 and up, iOS 7 and up, Windows 10 and Blackberry 10.

3.1.3. Unity

Unity 3D[13] engine is a tool to create games, applications and 3D animations. It is a multiplatform engine that allows you to deploy across major mobile, desktop, console, and TV platforms plus the Web.

The Unity editor is a complex visual editor to develop games. A game in Unity is structured in scenes that can be any part of the game, from a start menu to a level of your game or an area of al level. Unity engine also includes a terrain editor where you can create the terrain of your game using visual tools, painting, etc.

In Unity editor you can create your own objects or you can import them from different 3D platforms like Blender, 3ds Max, SketchUp, Maya and many more. Unity can read 3D formats like ".FBX", ".OBJ" or ".dae" files so you can import 3D objects from any platforms that can export to these formats. Unity can also read proprietary 3D application files like ".Max", ".Blend" or ".skp".

The behaviour of a Unity game object is defined by his scripts -one or more- that you can develop in JavaScript or C#. All the physics and many other behaviours are also provided from the Unity engine.

Although you can also create your own methods and classes all Unity scripts derive from the base class "MonoBehavior". This class provides some facilities to interact objects with unity engine including same methods to initialize whatever you want even before the scene of your game starts. The behaviour of objects

[12] Ionic official site: https://ionic.io/
[13] Unity 3D official site: https://unity3d.com/

is mainly based on frames, that is, the script is continuously repeated frame by frame. Same methods are called every frame or even more per frame. After initialization you can control physics, input events, game logic or even the rendering of the objects.

Unity 3D is a 3D game engine officially launched on June 2005. This engine allows to create games and other interactive content or 3D animations in real time. Many people interested in development find the difficulty of learning programming languages and the engines that use them. The creation of video games become very difficult without programming studies or computer animation skills.

Unity Technologies is one of the companies that has decided to change this situation. The Unity development team has decided to maintain the source code but offering the user a complete graphical interface so that users can control the source code without creating new elements in the code. This is what has made Unity so popular for video game developers.

Unity is a 3D real time and multimedia application besides being a 3D and physics engine to create games, animations in real time with audio, video and 3d objects content.

This engine does not allow complex modelling but you can create scenes with illumination, terrains, cameras, textures, etc. Unity allows to deploy video games to Windows, Mac OS, iOS, Android, Wii, PlayStation, Xbox 360, Nintendo, Web and same TV platforms.

Unity has several advantages that make it one of the most used video game engines at this moment. Some of these advantages are:

1. Allows to import several 3D formats like 3ds Max, Maya, Blender, SketchUp, FBX, etc. and you also can import some resources like Photoshop textures, PNG, TIFF, audios and videos.
2. It is compatible with graphic Direct3D, OpenGL and Wii graphic APIs. It is also compatible with QuickTime and use internally Ogg Vorbis format.
3. In Unity, the game is built using the editor and a scripting language so user does not have to be a programming expert to create a video game. Scripts can be created in JavaScript or in C# language.
4. The Unity game structure is defined by scenes that represent some part of the game.
5. Includes a terrain editor that allows you to create a terrain from the beginning, making his geometry, texture and including 3D elements that you can import from other 3D applications or elements already predefined in Unity.
6. There are several Unity versions but the simplest is free for personal use and let you create commercial games till you get $100k per fiscal year.
7. As we said Unity is multiplatform, so you can easily deploy on multiple mobile platforms, desktop, etc.

3.2. Cloud services technologies

Some of the services are not only composed by a TV app, but also a cloud backend or a web or a mobile app are part of the service. The main objective of cloud backend applications is managing the users' data for each service. The backends are not integrated in only one SENIOR-TV cloud application, but each service will access to its own cloud. The resulting architecture is similar then microservices[14] approach (see section 4. SENIOR-TV Platform).

Different technologies were used to developed the cloud backend services: Java/Spring, .Net, REST, etc. The flowing paragraphs introduce the most relevant characteristics of them.

3.2.1. Spring framework

Most of the SENIOR-TV cloud services are based on Spring Framework[15] technology (Agenda, Audiovisual Channels, Tracker, Video chat and Weather) combined with other third-party services.

The Spring is an open source application framework and control container for Java EE[16,17] platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java Enterprise Edition platform.

The Spring framework is a layered architecture which consists of several modules which provide everything that a developer may need for create an enterprise application. As modules are independent, developers can select the features that they need and eliminate the non-used the modules. Next figure shows the most relevant modules of Spring framework architecture:

[14] https://en.wikipedia.org/wiki/Microservices
[15] Sprint Framework web page: http://spring.io/projects/spring-framework
[16] Java EE reference: https://www.oracle.com/technetwork/java/javaee/documentation/index.html
[17] Java developers web page: https://www.oracle.com/technetwork/java/index.html

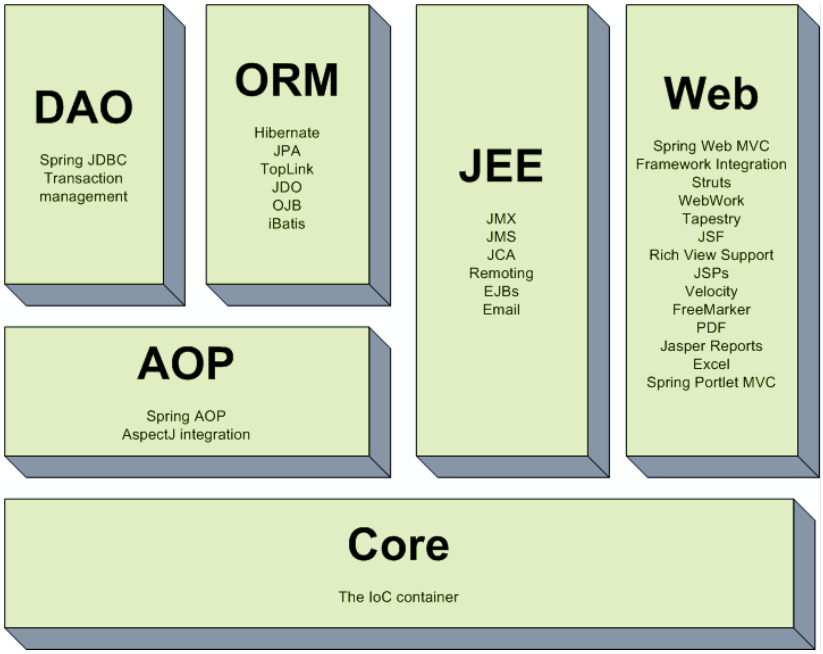


FIGURE 3. SPRING ARCHITECTURE

- **Core Module**: is the base framework module which provides the Spring containers, allows the application components configuration and manage the Java objects life cycle through dependency injection.
- **AOP Module**: this module allows developers to define interceptors and cut points to keep the concerns apart. It is configured at run time and aims at declarative transaction management easy to maintain.
- **DAO Module**: is the responsible of creating database connections and uses AOP to manage transactions. Spring works with relational database management systems on the Java platform using Java Database Connectivity (JDBC) and object-relational mapping tools and with NoSQL databases (transactions, DAO support, JDBC, ORM, Marshalling XML).
- **ORM Module**: integration with popular Object Relational mapping tools like Hibernate, iBATIS SQL Maps, Oracle TopLink and JPA etc.
- **JEE Module**: provides support for JMX, JCA, EJB and JMS etc. In lots of cases, JCA (Java EE Connection API) is much like JDBC, except where JDBC is focused on database JCA focus on connecting to legacy systems.
- **Spring MVC**: an HTTP servlet-based framework which provides tools in order to create web applications and RESTful web services. It also integrates well with other MVC frameworks like Struts, Tapestry, JSF, Wicket, etc.

3.2.2. .Net

.NET Framework is a software framework developed by Microsoft to run on Microsoft Windows. It includes a large class library "Framework Class Libraries" (FCL) which provides language interoperability across several programming languages.

FCLs provide a lot of functionalities such as user interface, database connection, data access, authentication, web development, or network communications. Developers combine their code with .NET Framework libraries in order to create the applications. Microsoft also provides an integrated development environment for .NET called Visual Studio. The architecture layout of the .NET Framework is illustrated in next figure:

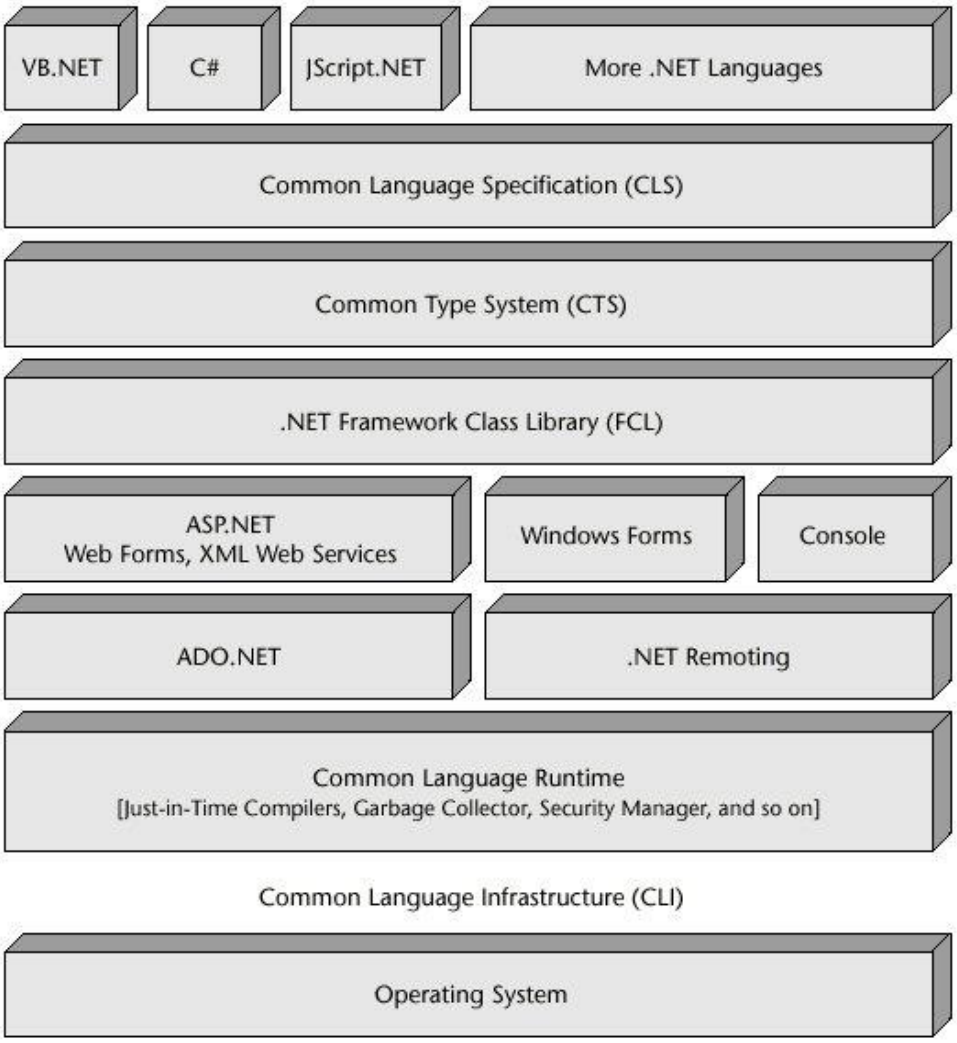


FIGURE 4. .NET COMPONENTS

- **Common Language Infrastructure** (CLI) provides a language platform for develop and execute software applications. By implementing the core aspects of .NET Framework within the scope of

CLI, these functions will not be tied to one language but will be available across the many languages supported by the framework.

- **Common Language Runtime** (CLR) serves as the .NET Framework execution engine and it offers many services such as memory management, type safety, exception handling, garbage collection, security and thread management. Programs written for .NET Framework are compiled into Common Intermediate Language (CIL) code. During execution, the CIL code is transformed into machine code.
- **Assemblies**: CIL code is organized in assemblies. The assemblies are stored in dynamic-link library (DLL) and executable EXE files. Each assembly consists of one or more files, one of which must contain a manifest bearing the metadata for the assembly.
- **Class library** implement many common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation. These libraries are organized in hierarchical namespaces.

3.2.3. REST

Representational State Transfer (REST) is a software architecture which defines a set of requirements for creating web services.

A web services is any functionality offered by a software system accessible by other systems via World Wide Web. This technology uses a set of standard protocols to interchange data among applications which can have been developed using different technologies and can be running over different platforms (mobile, web, etc.). The HTTP web technology is used by web services to transfer the information among applications, more specifically for transferring machine-readable file formats such as XML and JSON.

Web services that conform to the REST architectural style, termed RESTful web services, provide interoperability between computer systems on the Internet. RESTful web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations.

- **Stateless** means that the server does not need to know anything about what state the client is in and vice versa.
- Interact through **standard protocols** provides that systems not have to implementation any specific interface. If several applications hit the same REST endpoint, will perform the same actions, and will receive the same responses.

- Using **HTTP** standard protocol allow make service request easily, only need to indicate the resource path, a header, an optional message body containing data and the HTTP verb which defines what kind of operation to perform (GET, POST, PUT or DELETE).

3.2.4. Firebase

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. Since Google acquisition till today, Firebase expanded their services to become a unified platform for mobile developers, integrating several Google services like "Google Cloud Platform", "AdMob", and "Google Ads", between others, in order to offer broader and scalable products. The main services provided by Firebase are the following:

- **Firebase Analytics** is a cost-free app measurement solution that provides insight into app usage and user engagement.
- **Firebase Cloud Messaging** is a cross-platform solution for messages and notifications for Android, iOS, and web applications, which can be used at without costs.
- **Firebase Auth** provides a user authentication using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google. Additionally, it includes authentication with email and password mechanism.
- Firebase provides a **real time database** backend service which allows application data to be synchronized across clients and stored on Firebase's cloud. Realtime database is accessible by REST API technology.
- **Firebase Storage** provides secure file uploads and downloads (images, audio, video, or others) regardless of network quality.
- **Firebase Hosting** is a static and dynamic web hosting.
- **Firebase Performance** provides insights into an app's performance and the latencies the app's users experience.
- **Firebase Test Lab for Android and iOS** provides cloud-based infrastructure for testing Android and iOS apps. Test results (logs, videos, and screenshots) are available from Firebase console.

4. SENIOR-TV Platform

4.1. User interaction

SENIOR-TV is not just a TV application for elderly, but it is an opened platform for different types of users: seniors, relatives and caregivers.

The platform services are mainly focused on being consumed through a smart-TV. On the other hand, considering the different user types, some of the services are accessible through other types of devices such as smartphones, tablets or PC's. The next diagram shows the principal users of SENIOR-TV and how each kind of user will access to the platform.

FIGURE 5. PLATFORM INTERACTION

Elderly people will be the main users of SENIOR-TV services. The majority of time, they will access the platform through smart-TV. However, certain services such as “Tracker”, “Agenda”, or “Weather” will be accessible by other smart media (web, mobile or tablets).

Although secondary users (relatives and caregivers) can use their TV to access to the platform, they will normally use their smartphones or tablets to access to SENIOR-TV services.

In addition, some of the services like “Audiovisual channels”, “Virtual center” or “Weather” will have different functionalities depending on their TV, mobile or web app.

4.2. General architecture

The diagram below shows the architecture of SENIOR-TV platform and the technologies used to develop each component.

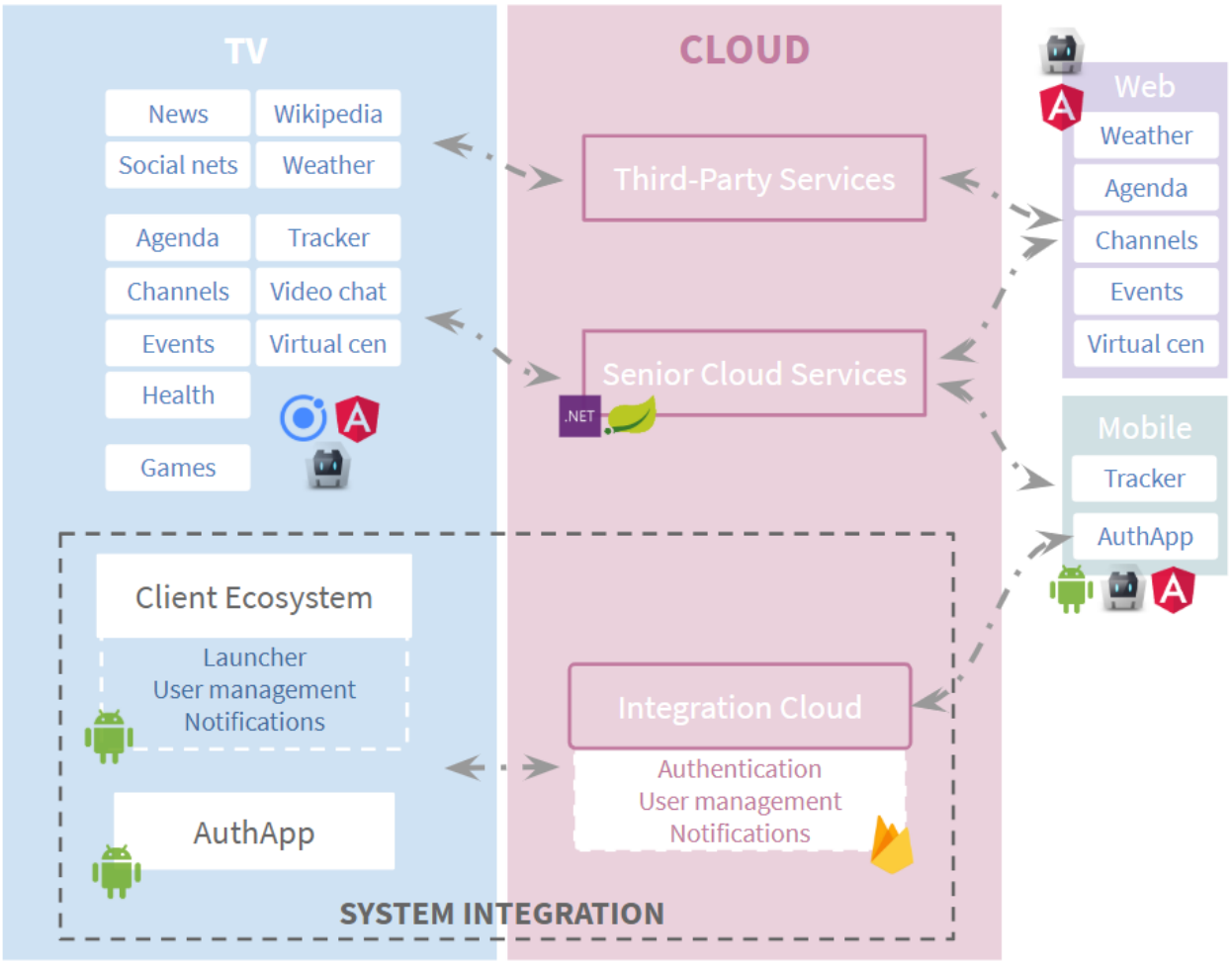


FIGURE 6. PLATFORM INTERACTION

The platform is composed by for main blocks:

- **TV** is formed by two kind of applications:
  1. Formal and informal TV apps which correspond with the SENIOR-TV services. The "Senior Cloud Services" and "Third-party Services" information are accessed by these apps through API REST requests.
  2. TV System Integration (Client Ecosystem + AuthApp) are the apps responsible of combine the user manage and the SENIOR-TV service functionalities and presenting them as a TV unified system.
- **Cloud** includes the services responsible to store and manage the user information, their devices and the use of the platform services. It is composed by three different kind of services.
  1. Cloud System Integration (Integration cloud) manages the user and devices information.
  2. Senior cloud services are the backend applications of some SENIOR-TV services. The web and TV apps will access to the database information using these services APIs.
  3. Third-party Services includes all the auxiliary services used by SENIOR-TV services such as weather forecast or news repositories.
- **Web**: the web app of some particular services (the "Senior Cloud Services" and "Third-party Services" information will be access by these apps through API REST requests).
- **Mobile**: the mobile apps of some particular services.

5. Informal services

The following subsections show the main characteristics and the informal services state after the three development cycles (months 1 to 35). The section shows the services evolution along of the three SENIOR-TV versions.

All the services are translated into the languages of the three pilot languages: Greek, Romanian and Slovenian. In addition of contents, the services names are localized following the trainers' suggestions.

5.1. SENIOR-TV V1

The platform obtained after the first cycle only includes services which run in the smart-TV devices, as the following chart is showing:

FIGURE 7. CURRENT APPS INTERACTION

The aim of the SENIOR-TV first version was developed the initial versions of a set of informal services to be tested during the first cycle of WP3 pilots by seniors from the three pilot countries: Cyprus, Slovenia and Romania.

That first version did not include complete functional services, but it included initial versions like clickable mock-ups, which were improved in successive versions using the feedback obtained from seniors during the pilots. In that moment, seniors evaluated issues such as the look and feel colours, the type and size of fonts, usability, the interaction devices and example contents.

The WP1 workshops provided valuable input from the users. Based on those, the services to be implemented in the first iteration were selected. According to those results, the services most demanded by the elderly, regardless of their sex and age, are: the ***weather***, the ***news*** and ***events***, apart from ***games*** and **entertainment** services in general.

5.1.1. Events 1.0

The scope of this service is to enable elderly to find and participate into events that they are interested in by just selecting an event on the TV screen.

The app is also localized, meaning that it will only show events accessible to the elderly user and in his own language. During the writing of this report we are also investigating the possibility of enabling users to participate to online events.

The following chart shows the architecture of the Events service:

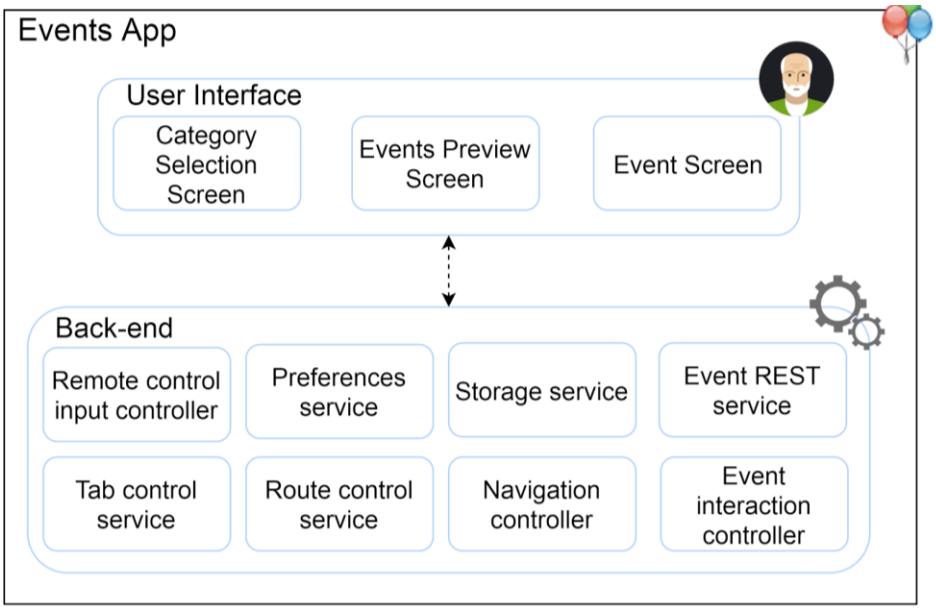


FIGURE 8. EVENTS ARCHITECTURE

The Back-end of the Events service has the following components:

1. Remote-control input controller: is the service that handles the user input and transform the input into actions inside the app.
2. Preferences service: is responsible for managing the preferences of the user (e.g. language).
3. Storage service: is responsible for storing information, locally on the device, about the interaction with the app and also about each session the user has with the app.
4. Tab control service: is responsible for managing the switching between different types of news categories.
5. Route control service and Navigation Controller: allow the user to navigate from one page of the app to another page of the app with ease.
6. Event Interaction Controller: is responsible for enabling the user to interact with a specific event. The user will be able to send a notification for Participation to the Event Organisers.
7. Event REST service will be responsible for fetching and parsing different event metadata (e.g. location, date, availability, price, etc.).

The main screens of the TV interface are as follows:

1. The Category Selection Screen

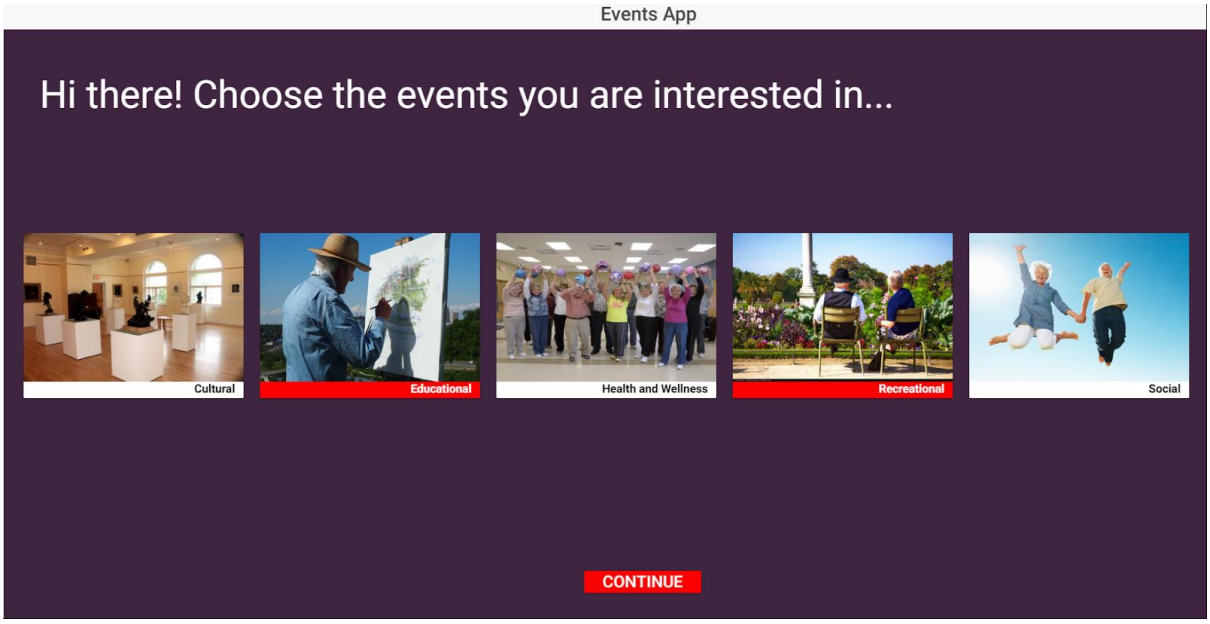


FIGURE 9. EVENTS APP SELECTION SCREEN

2. The Events Preview Screen

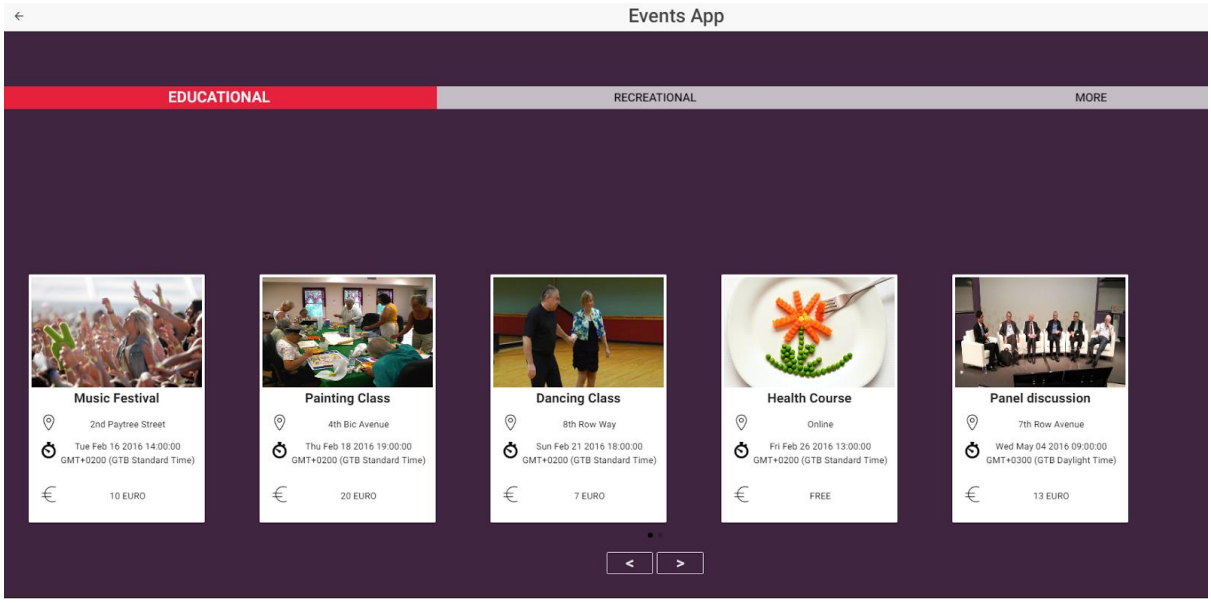


FIGURE 10. EVENTS APP PREVIEW SCREEN

3. The Events View Screen

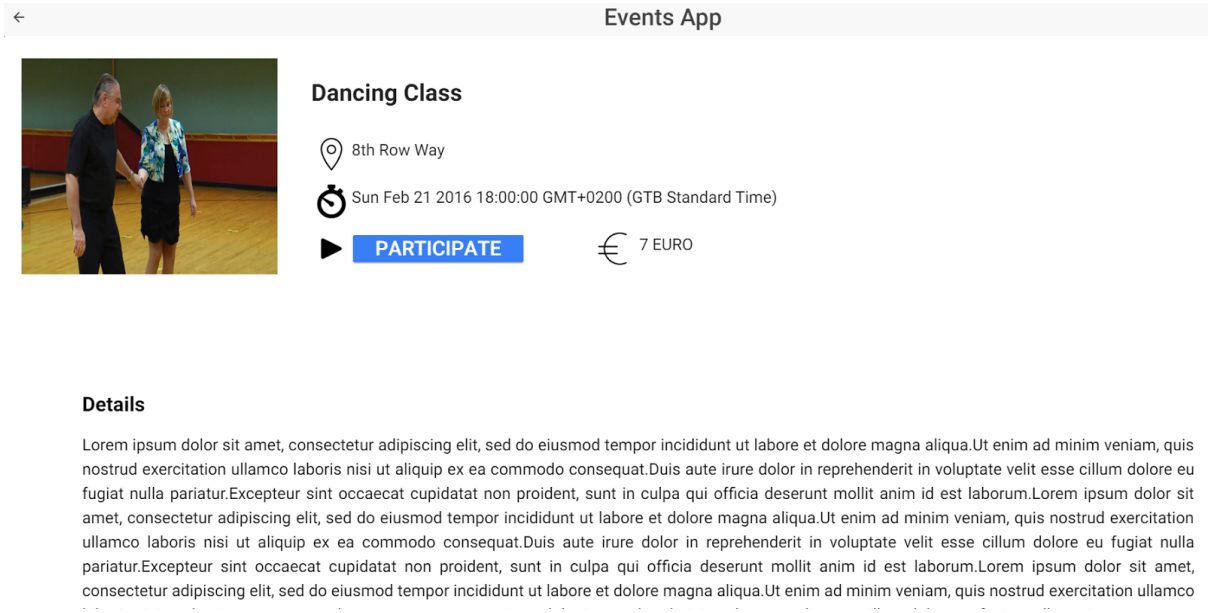


FIGURE 11. EVENTS VIEW SCREEN

5.1.2. News 1.0

The News service scope is to facilitate the viewing of news in an easily manner on TV, while also taking into consideration personal preferences and interests. The elderly user is able to choose which category of news he wants to follow and which news to read.

The app runs directly on the Set Top Box and fetches the News from different RSS News Websites. The app and news feeds are localized, meaning that the users from different countries see the news from their own countries and in their own language.

The next diagram shows the architecture of the News app:

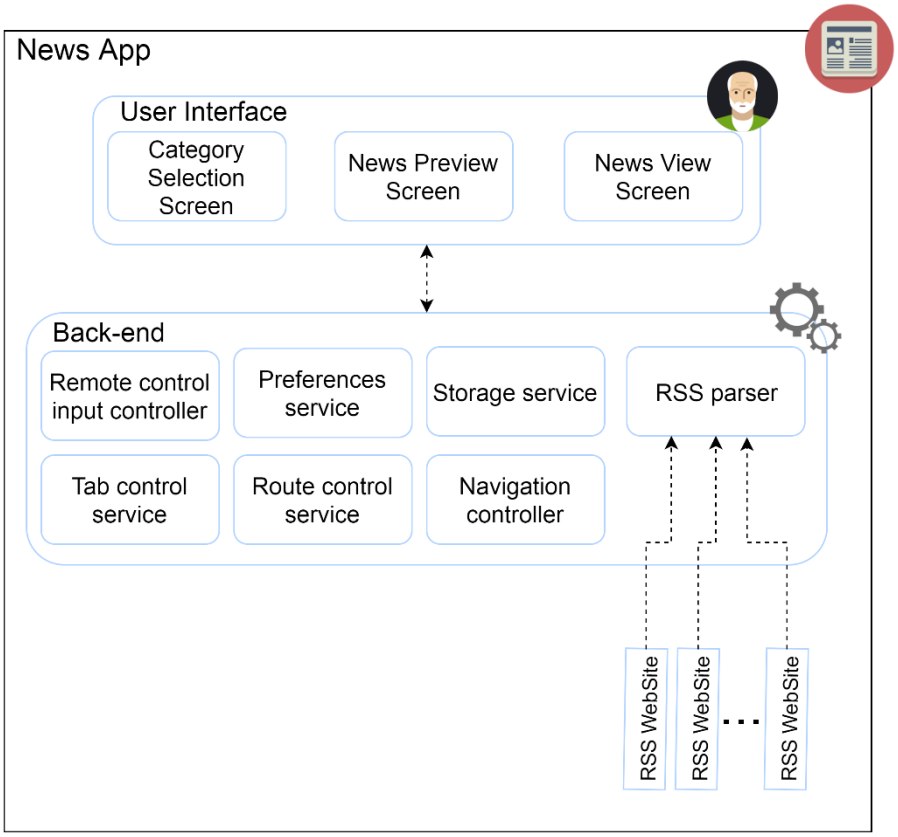


FIGURE 12. NEWS ARCHITECTURE

The back-end of the News app has the following components:

1. Remote-control input controller: is the service that handles the user input and transform the input into actions inside the app.
2. Preferences service: is responsible for managing the preferences of the user (e.g. language).
3. Storage service: is responsible for storing information, locally on the device, about the interaction with the app and also about each session the user has with the app.
4. RSS parser: is in charge of fetching news from RSS feeds and parsing them for presentation on the user interface.
5. Tab control service: is responsible for managing the switching between different types of news categories.
6. Route control service and Navigation Controller: allow the user to navigate from one page of the app to another page of the app with ease.

The application has one screen for viewing the news information and two main screens for browsing the news and the preferences:

1. News Central Screen

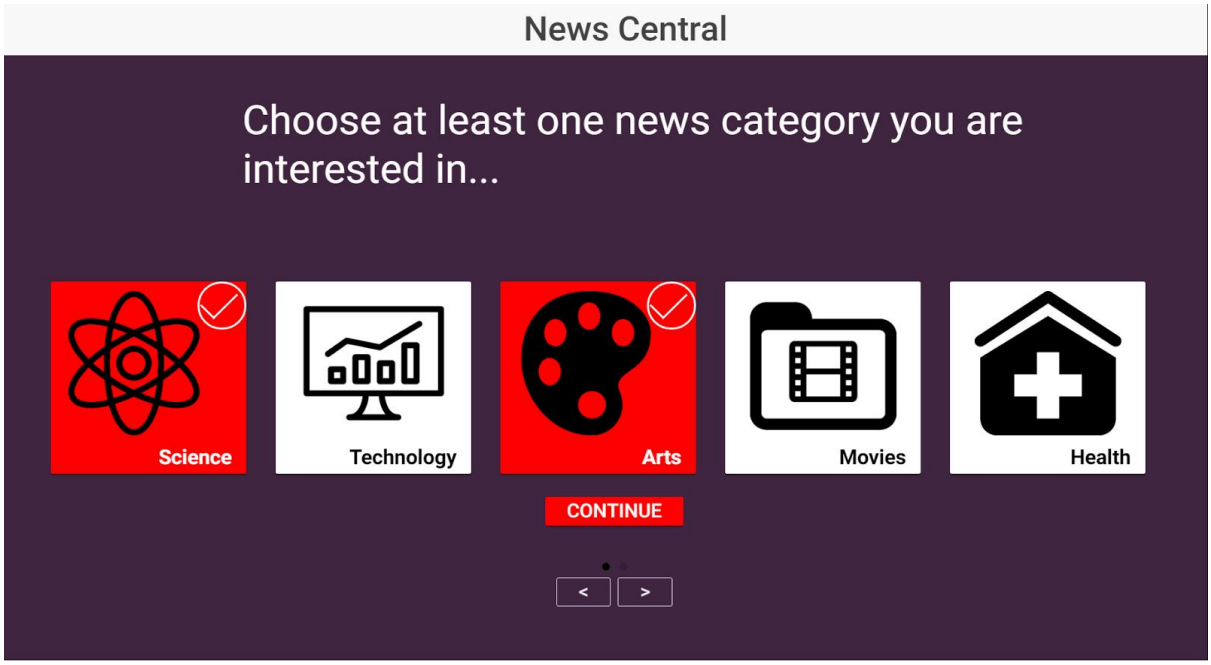


FIGURE 13. NEWS APP CENTRAL SCREEN

2. Category List News Screen

FIGURE 14. CATEGORY LIST NEWS SCREEN

5.1.3. Weather 1.0

The Weather service allows seniors to check the weather conditions for their village or city. The elders don't need to select their location, the app is able to get the users' position and ask the forecast automatically.

The app runs directly on the Set Top Box and fetches the weather data via HTTP requests to an open weather API free service, called *OpenWeatherMap*[18]. The Weather app are localized, meaning that the users from different countries see the information in their own language.

The next chart shows the architecture of the Weather service:

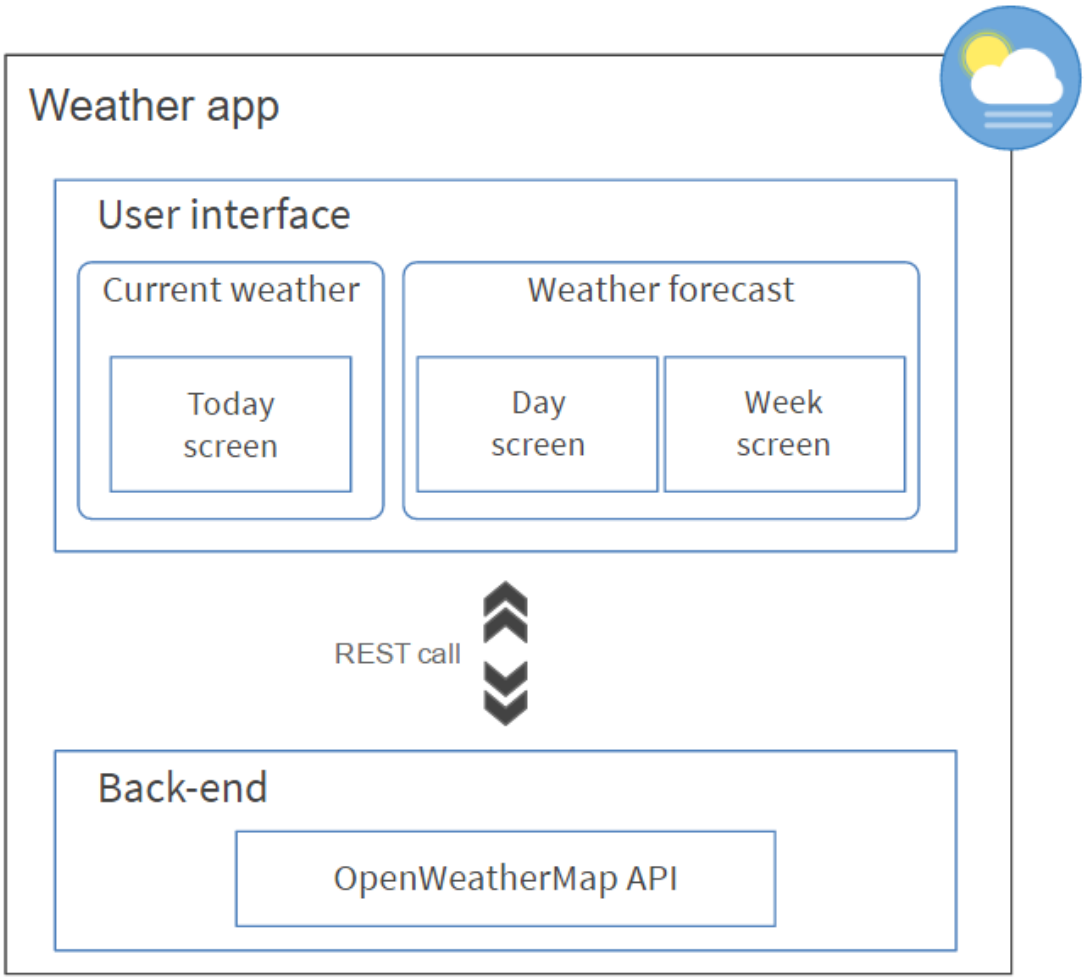


FIGURE 15. WHEATEAR APP ARCHITECTURE

The Weather interface has only one screen which is divided in two sections:

1. **Left** side shows the **current** weather conditions.

[18] OpenWeatherMap official site: https://openweathermap.org/

2. **Right** side shows the **forecast** for the current day or for the next three days.

1. Current weather

   Shows the temperature, humidity and wind forecast for the senior location, in this moment.

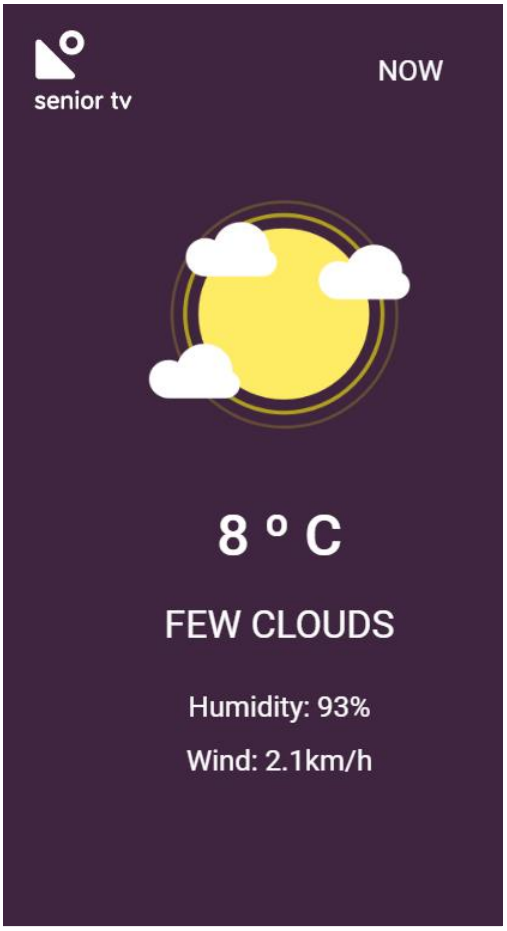


FIGURE 16. WEATHER APP CURRENT SCREEN

2. Forecast screen

   Shows the weather forecast for the current day, or for the next three days if we choose so. The user can navigate between these options using the top of the screen buttons: “Day” and “Week”.

   The user can switch the screens pressing the corresponding button or using the right and left key arrows of the remote-control.

   - Day screen

   In the case of the current day state, the information shown consists in a line chart with the temperature forecast for the next hours.

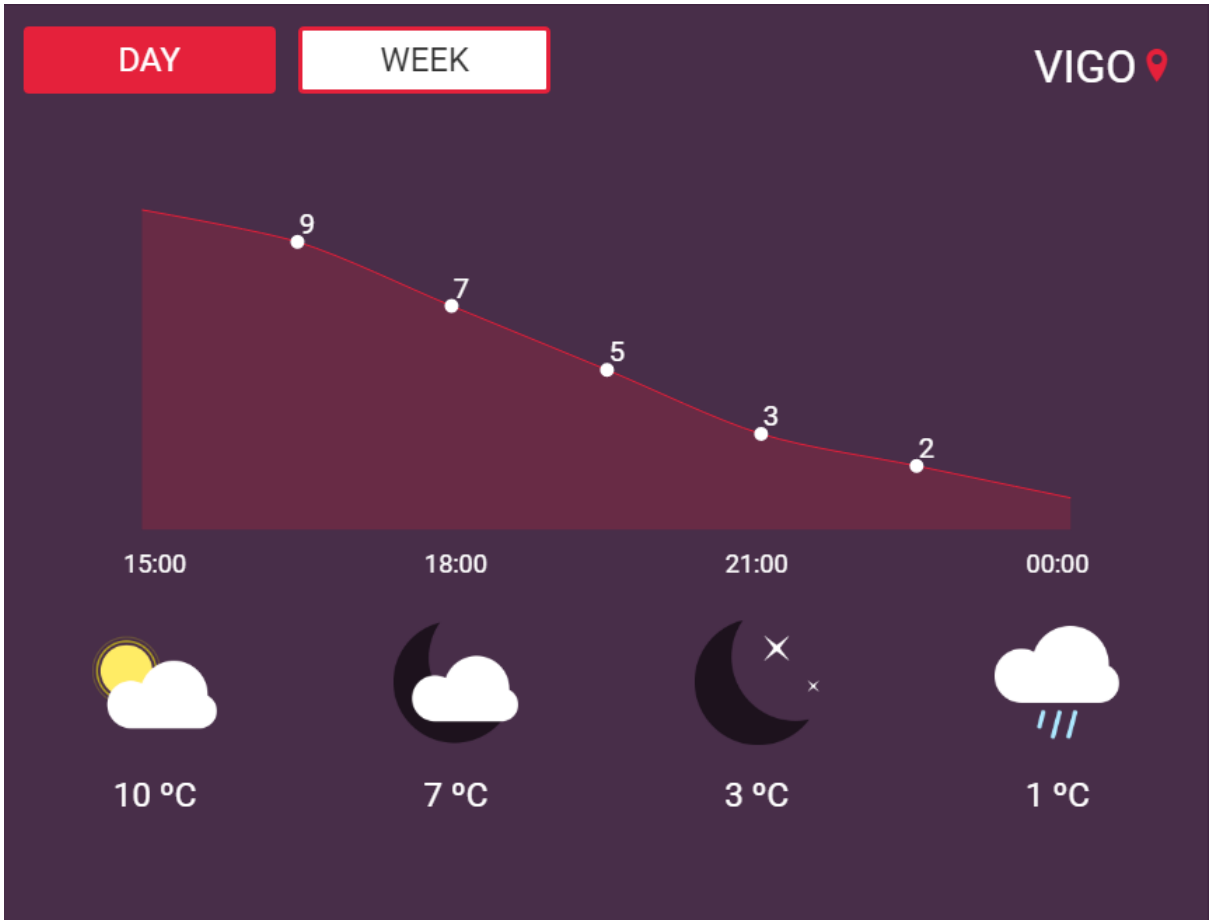


FIGURE 17. WEATHER APP DAY SCREEN

- Week screen

Instead, for the week state the application shows the weather forecast and the max and min temperatures forecast.

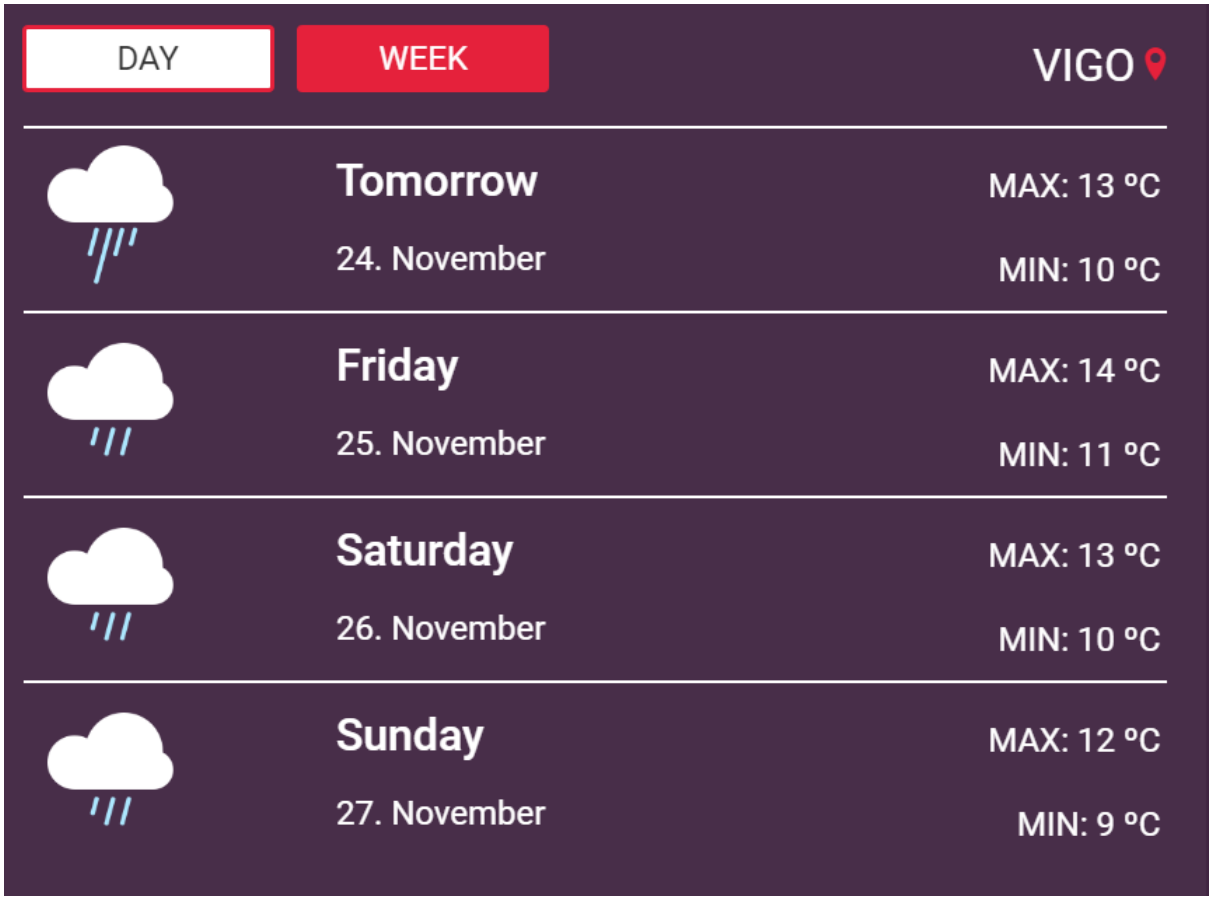


FIGURE 18. WEATHER APP WEEK SCREEN

5.1.4. Attentix

Attentix[19] is a game for improving memory skills. The game creates sequence of lights and asks the user to repeat it. If user do well, the game will increase number of lights in the sequence one by one. After that, asks a new sequence to the user. At the end, when the series generated users can always reach their maximum skill level in the game. Different levels can be selected. Each level adds new colour box in the screen. The next diagram shows the architecture of the Attentix game:

[19] The Attentix game is co-financed within the SENIOR-TV project.

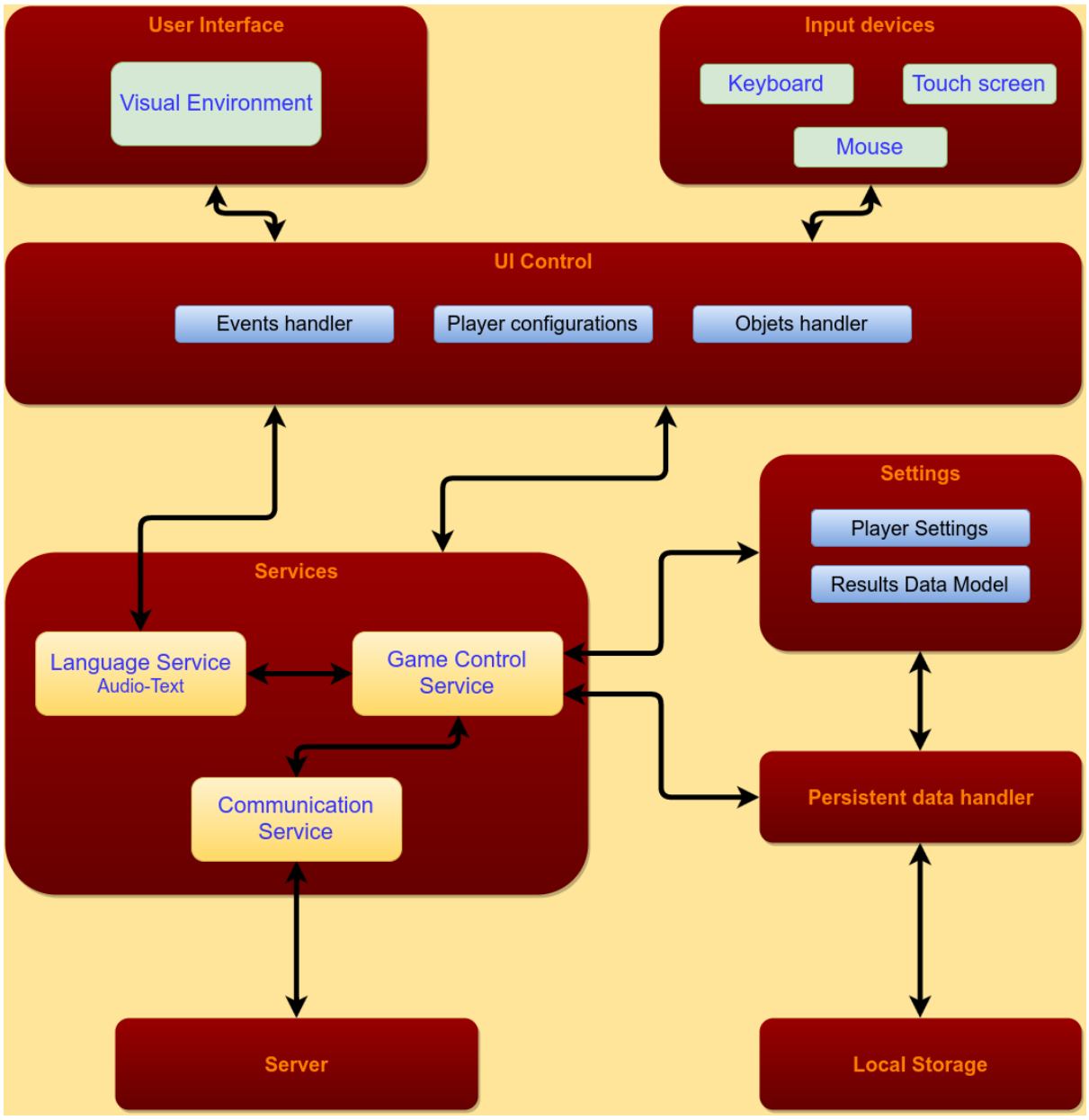


FIGURE 19. ATTENTIX GAME ARCHITECTURE

The previous chart shows the modules of the game and the relationships between them.

1. The Game Control Service module is the core of the game. It is used by other services or controllers to get current game information or to set it. Game Control Service has some different objectives like read current player setting to set the language or to prepare the results to send to the server.
2. The User interface block are those parts of the game responsible for the rendering of the game objects. This block is connected to the Game Control Service through the UI Control block to get object's positions for example.

3. The UI Control is used to interact between user interface and the input. Also, it is responsible for connecting with the Language Service to get the current translation for the words or sentences to show on screen. This block has to create the sequence to represent and control if is correctly pressed.
4. The Settings block is to represent the data model to save player setting or how to send results to the server.
5. The Communication Service is responsible for sending the game results to the server.
6. The Persistent data handler is the block responsible for storing results or player settings into the device. The data is saved in a persistent data folder with the package name.

Seniors have to use the pointer to interact with the game, clicking different squares on the screen. The next image shows the main screen of Attentix game:

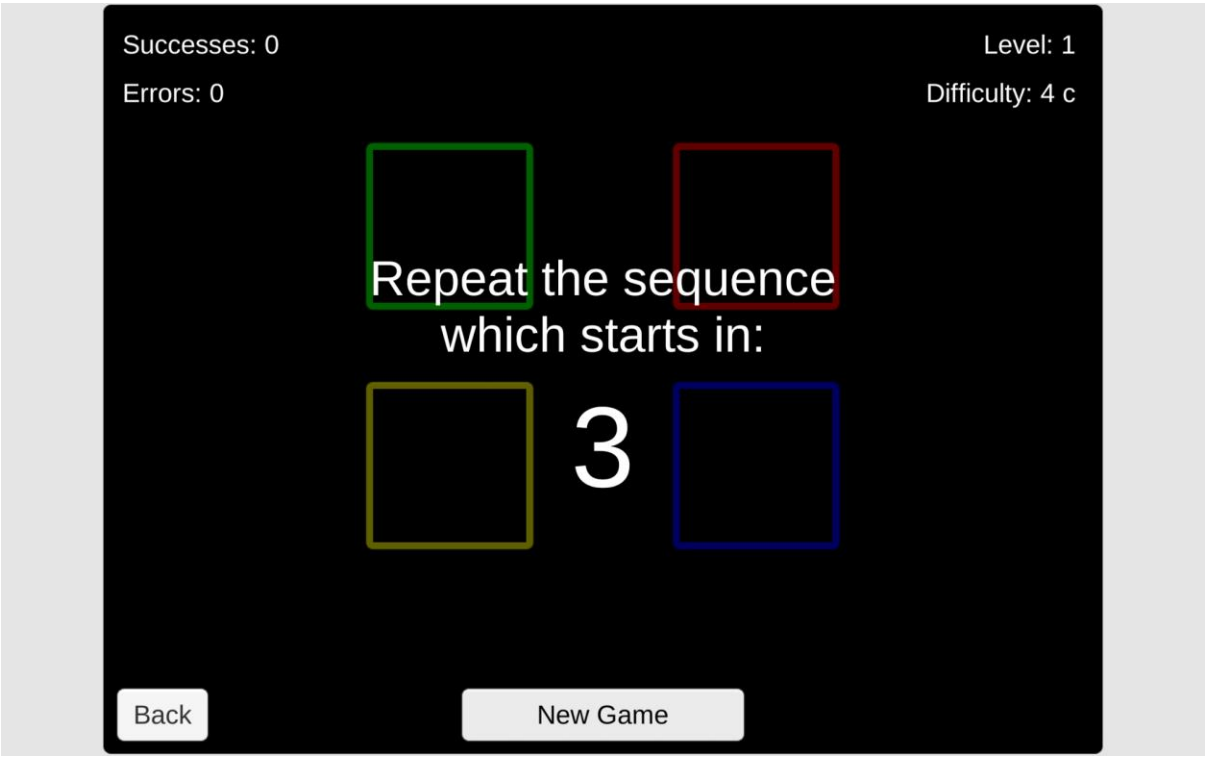


FIGURE 20. ATTENTIX GAME SCREEN

5.1.5. Episodix

Playing to Episodix[20] game, seniors can see a virtual city and walk around their streets. During their walk, some objects could appear and the senior have to remember which objects have appeared and which not. This game stimulates the elders' memory. The next chart shows the architecture of the Episodix game:

[20] The Episodix game is co-financed within the SENIOR-TV project.

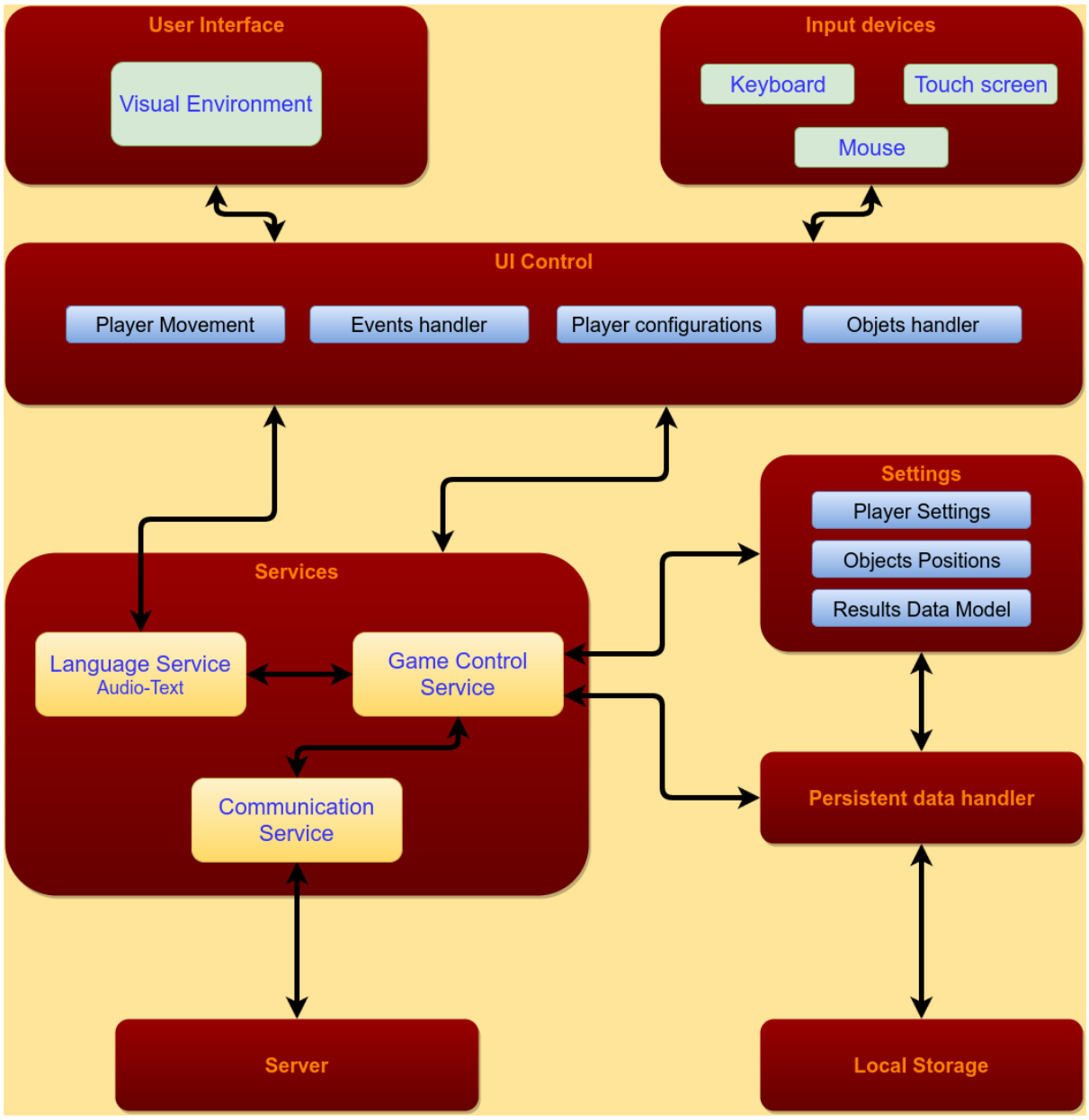


FIGURE 21. EPISODIX GAME ARCHITECTURE

1. The Game Control Service is the core of the game. It is used by other services or controllers to get current game information or to set it. Game Control Service has some different objectives like read current player setting to set the language or to prepare the results to send to the server.
2. The User interface block includes the parts of the game which are responsible for rendering the game objects. This block is connected to the Game Control Service using the UI Control block to get object's positions.

3. The UI Control is used to interact between user interface and the input. Also, it is responsible for connecting with the Language Service to get the current translation for the words or sentences to show on screen. The player movement is controlled in this block.
4. The Settings block represents the data model for saving the player settings.
5. The Communication Service is responsible for sending the game results to the server.
6. The Persistent data handler is responsible for storing results and the player settings into the device. The data is saved in a persistent data folder with the package name in the devices SD-card

Seniors have to use the pointer to interact with the game, selecting the up, down, left and right with arrows that appear on the screen. Following you can see the main screen of Episodix game:

FIGURE 22. EPISODIX GAME SCREEN

5.2. SENIOR-TV V2

After first pilot finishing, some valuable feedback was obtained from seniors. All that information was included in the services of the platform second version.

Second version included complete versions of previous services and four more services which objective was promote active lifestyle, seniors' autonomy and keep them informed. In addition, some of the services were extended to use other platforms: mobile, tablet and web.

5.2.1 Agenda 1.0

The Agenda service allows seniors, family members and caregivers to keep a record of seniors' appointments and events. Seniors may have trouble remembering all appointments on the day. This service can help them to organize their daily tasks, keeping them informed about the most relevant events (e.g. medicines intake, appointments with the doctor). Family and caregivers can also add events and reminders to seniors' agendas.

The Agenda service is composed by two different apps: a TV and a Web app. The TV app is focussed on seniors and the Web app is oriented to secondary users. The following chart shows the architecture of the service:

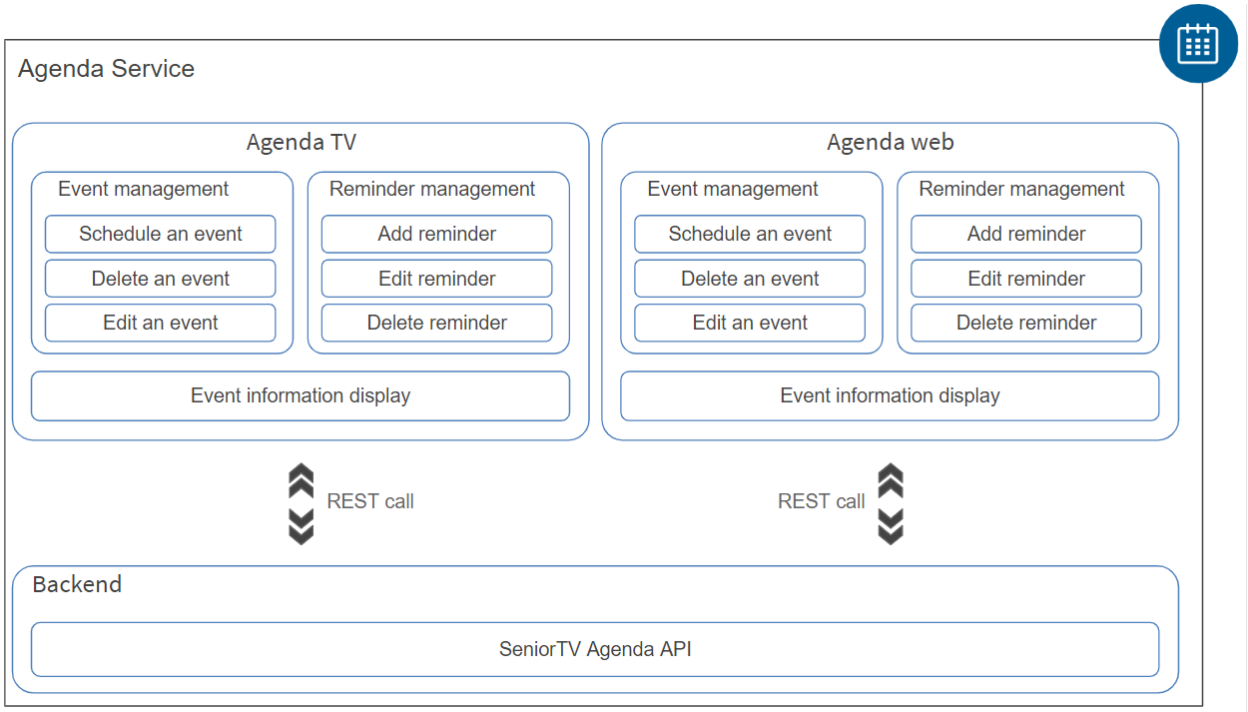


FIGURE 23. AGENDA ARCHITECTURE

Both, Web and TV applications have the same functionalities described below. The functionalities are arranged into two groups:

1. Events management: these functionalities allow users to read, create, update and remove Agenda events.
    - *Display an event information* allows users to read the information of a scheduled event: the Name or Title of the event, the Starting and Ending time, the Location or Address, the associated Reminder and the Reminder time.

- *Schedule an event* allows users to schedule an event. In particular, users can schedule an event with the following information: Name or Title of the event, Starting and Ending time, Location or address and, Reminder.
- *Delete an event* allows users to remove a scheduled event. To remove the information a user confirmation is mandatory.
- *Edit an event* allows users to modify the information of a scheduled event. In particular, the user can update the following information: Name or Title of the event, Starting and Ending time, Location or address and, Reminder.

2. Reminders management: these functionalities allow users to manage the reminder associated with an event.
   - *Add a reminder* allows users to set a reminder to an event. There are two ways of adding a reminder: when scheduling a new event or when modifying an event. Possible reminder values are: Never, 30 minutes before the starting time, 1 hour before the starting time or 1 day before the starting time.
   - *Edit the reminder* allows users to modify the event associated reminder. This functionality is only accessible when the user is updating an event.
   - *Delete the reminder* allows users to remove the event associated reminder selecting the option "Never".

The Agenda TV and Web have been implemented using Angular and Apache Cordova frameworks. The cloud backend has been developed using Java EE technology. The two apps use HTTP requests to communicate with the backend to obtain the seniors' data. The following paragraphs describe the TV and Web applications architecture and the main interface screens.

5.2.1.1 Agenda TV

The following charts show the Agenda TV app architecture and some interface screens:

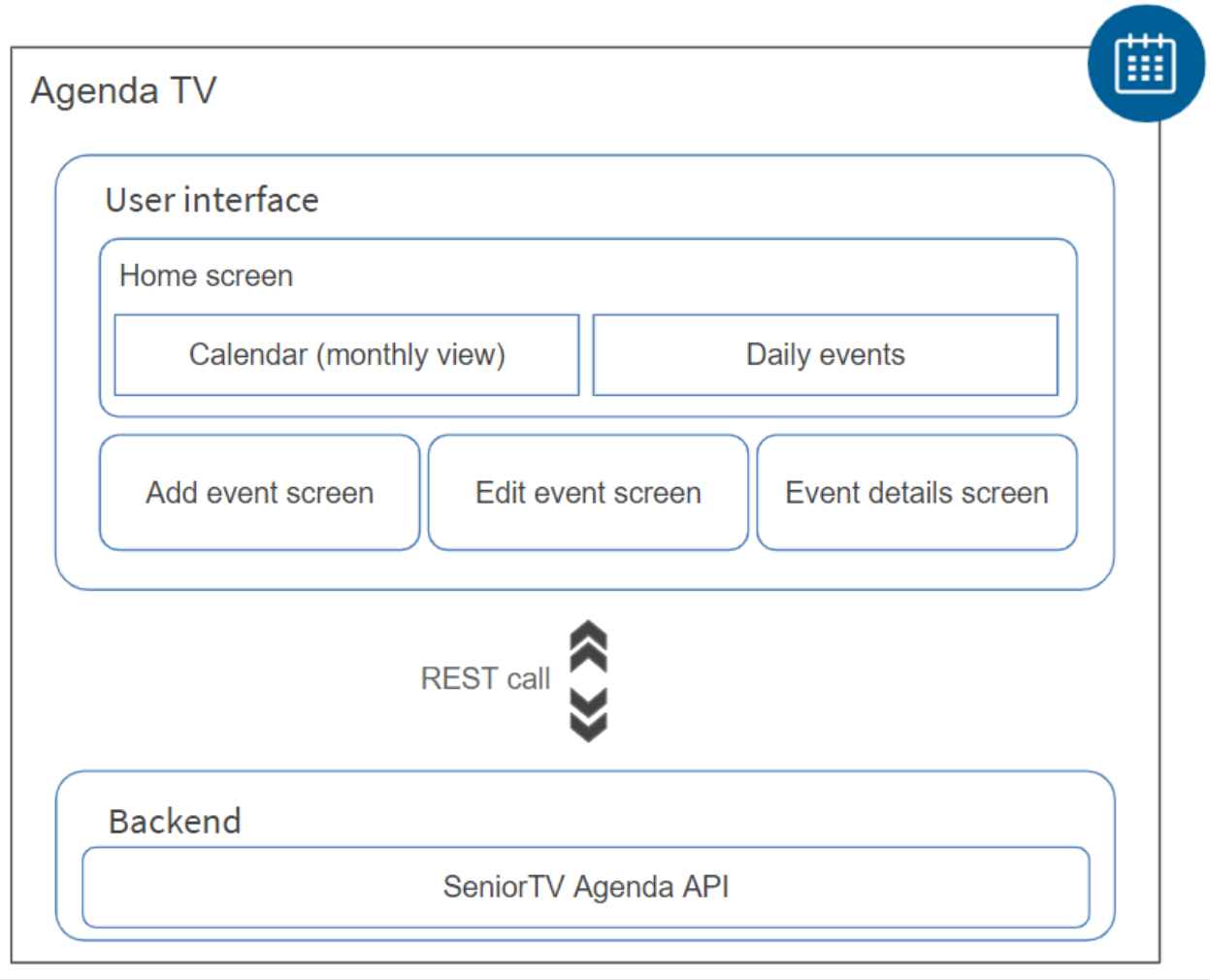


FIGURE 24. AGENDA TV ARCHITECTURE

1. Home Screen: The home screen is the first screen shown after the Agenda TV application starts. This screen is divided in two sections:
    - On the left side, the monthly view of the calendar: offers a general overview of the scheduled events for the current month. Days with scheduled events are highlighted in grey color, the current day is highlighted in red and the selected day is highlighted in orange.
    - On the right side, the scheduled events for a particular day are shown. Also, a "Add New Event" button is included to move to the "Add Event" screen to schedule a new event for the selected day.

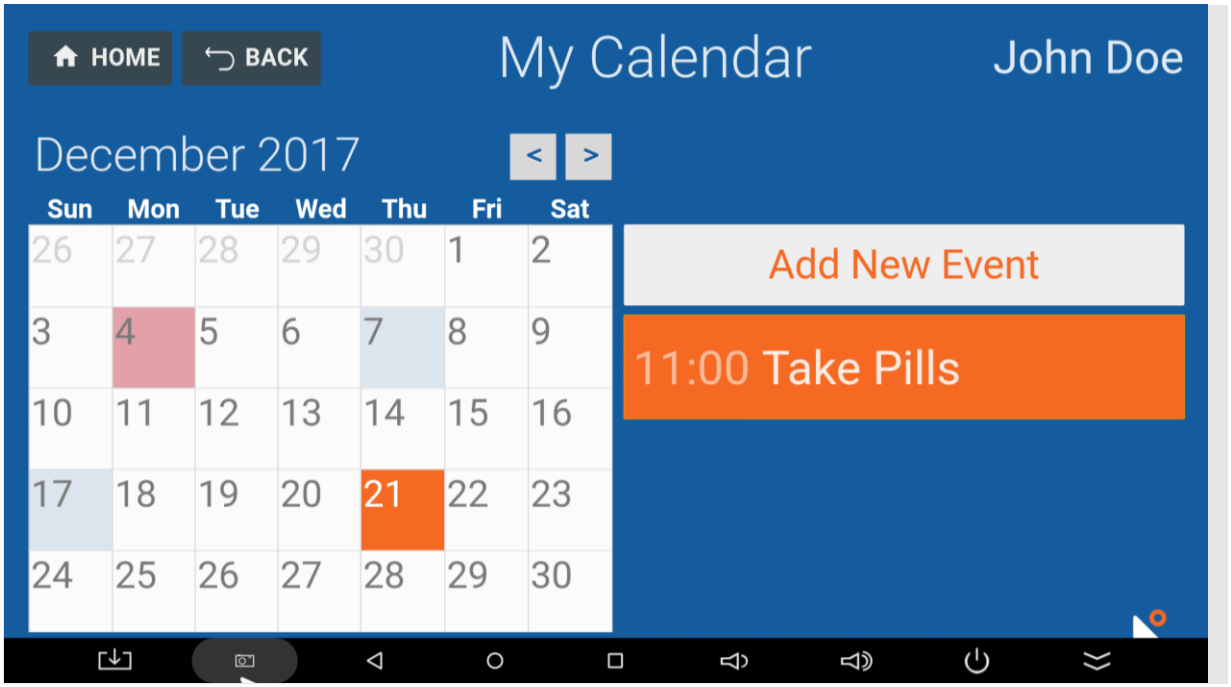


FIGURE 25. AGENDA TV HOME SCREEN

2. Event Details Screen

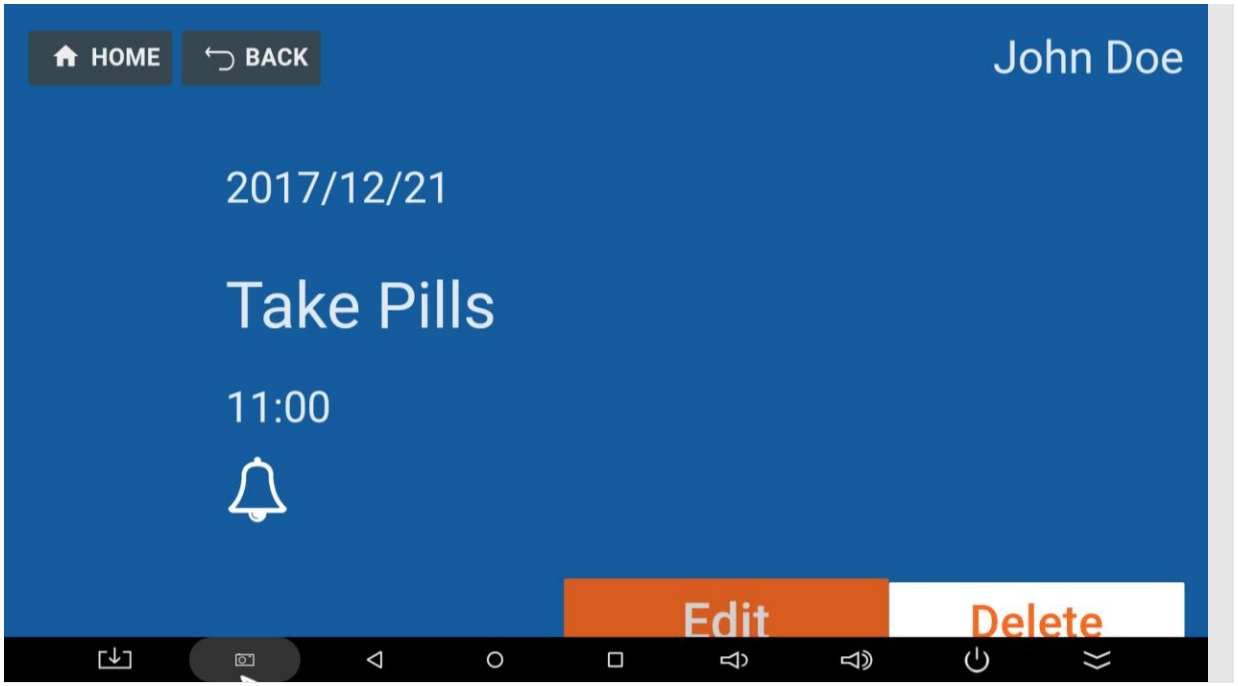


FIGURE 26. AGENDA TV EVENT DETAILS SCREEN

3. Add Event Screen

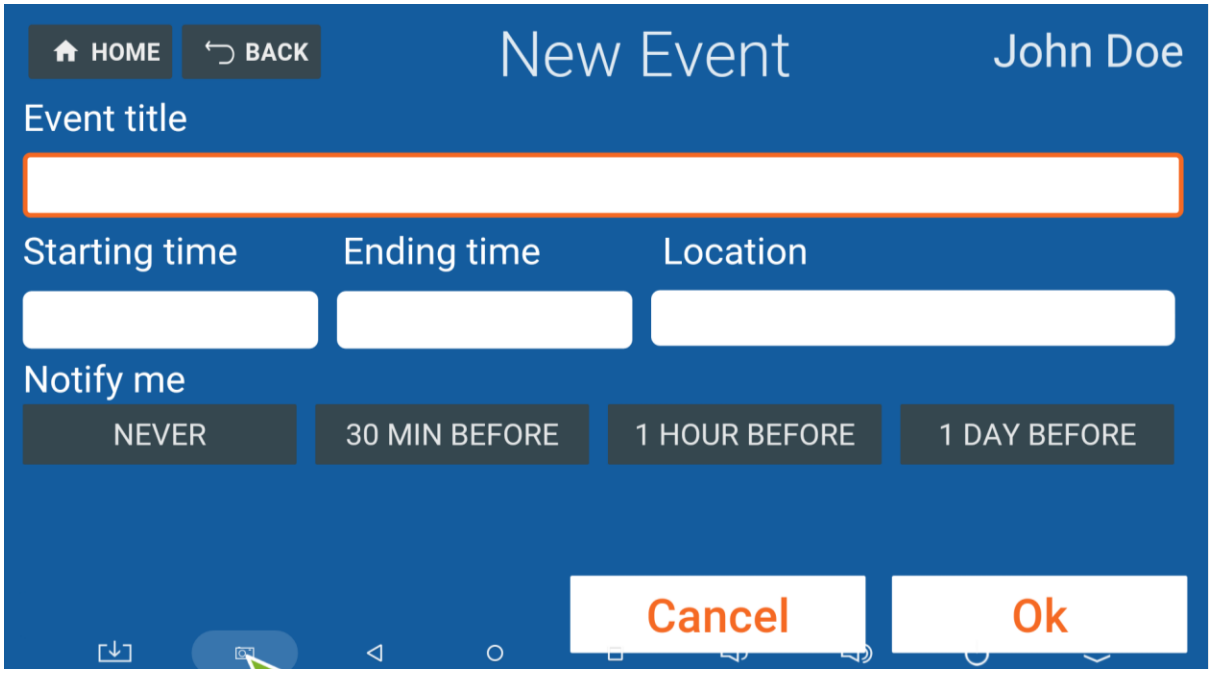


FIGURE 27. AGENDA TV ADD EVENT SCREEN

4. Edit event Screen

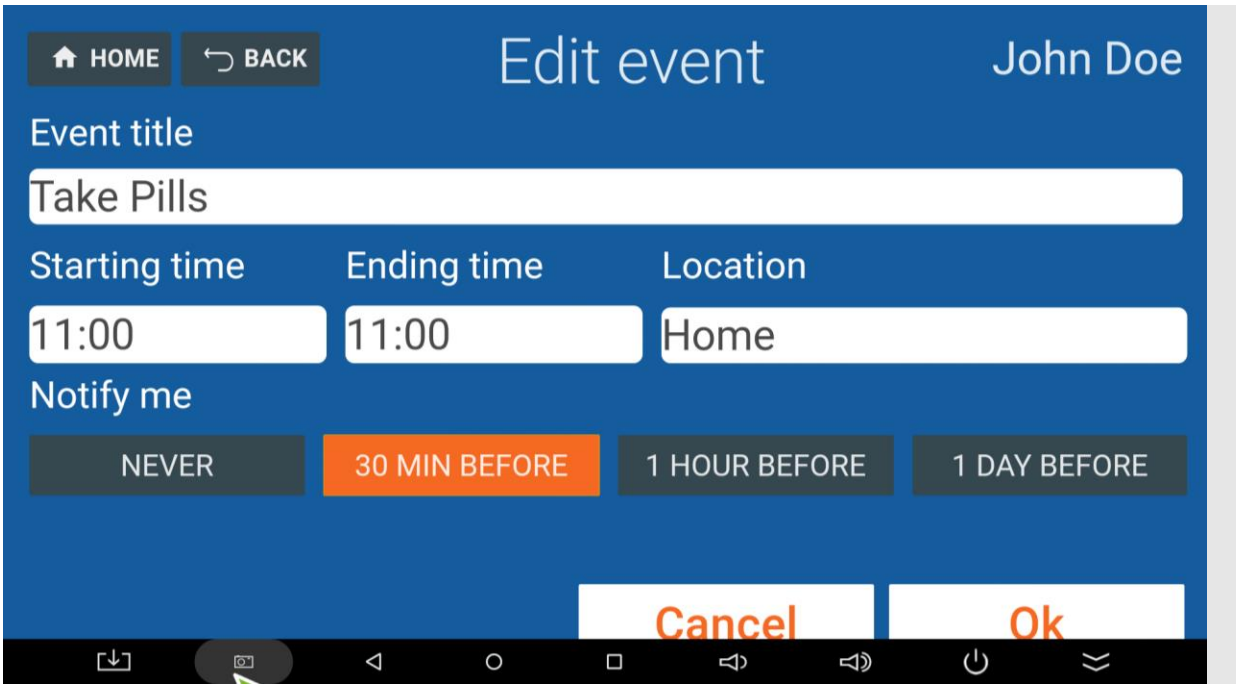


FIGURE 28. AGENDA TV EDIT EVENT SCREEN

5.2.1.2 Agenda Web

The Agenda Web app is accessible from this URL: https://www.imatia.com/seniortv-agenda-web/ The following chart shows the architecture of the user interface of the Agenda Web app.

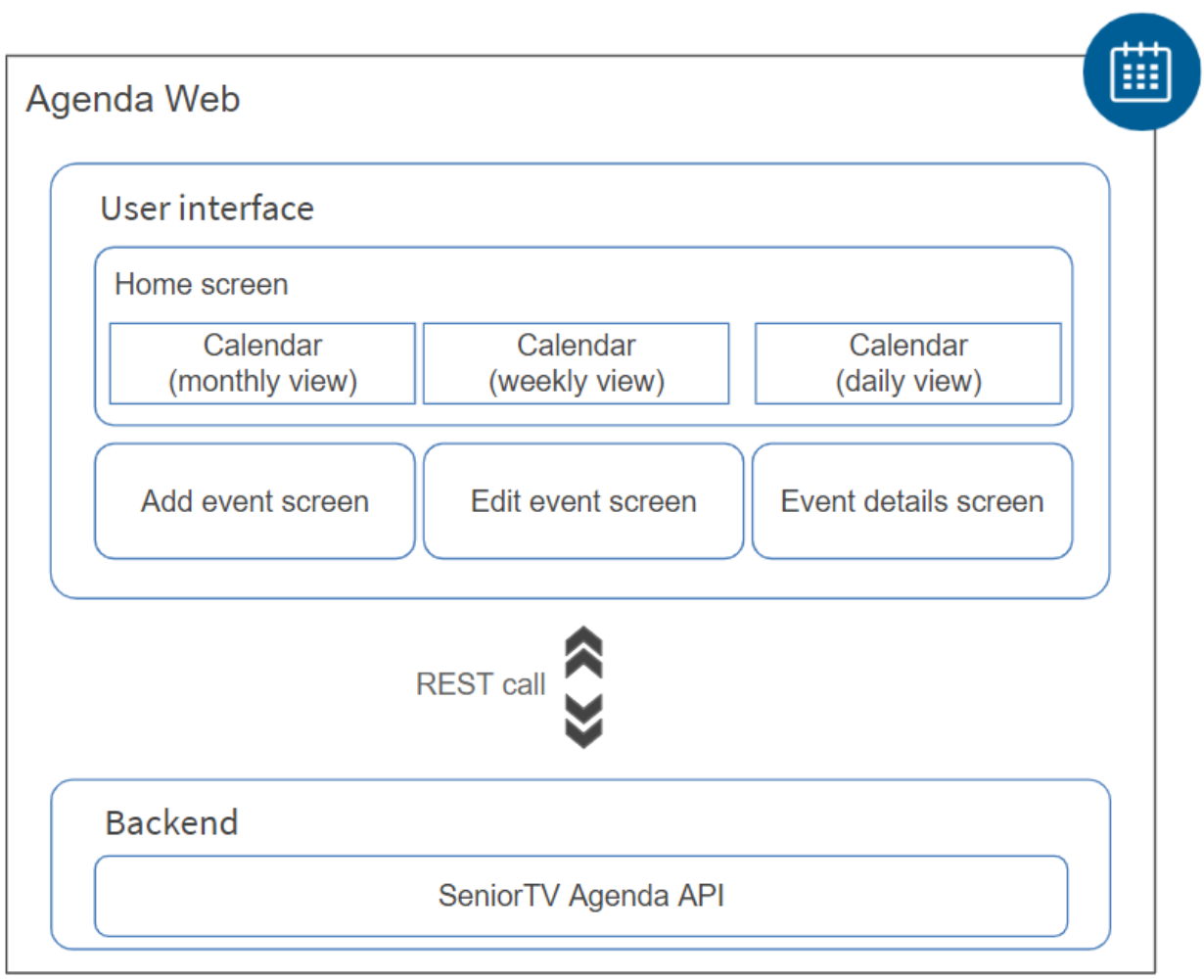


FIGURE 29. AGENDA WEB ARCHITECTURE

1. Home Screen is the first screen shown after the Agenda web application starts. By default, the application displays the calendar monthly view, which offers a general overview of the scheduled events for current month. The app also offers a Weekly and Daily view to filter the events.

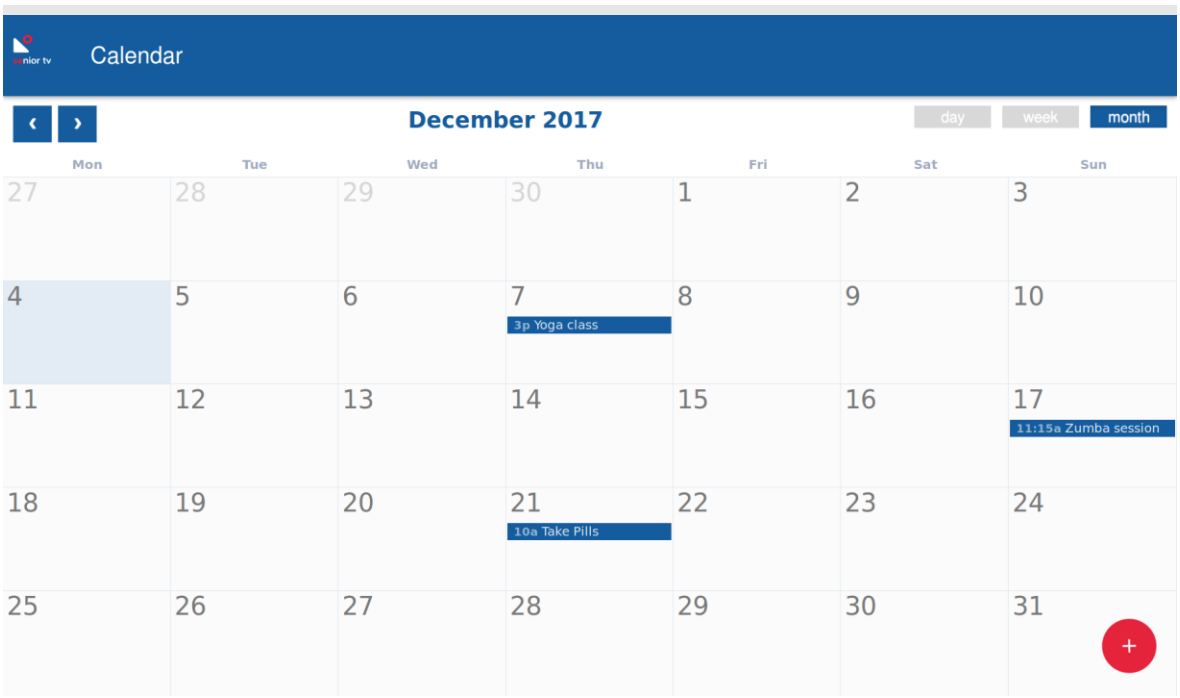


FIGURE 30. AGENDA WEB HOME SCREEN MONTHLY VIEW

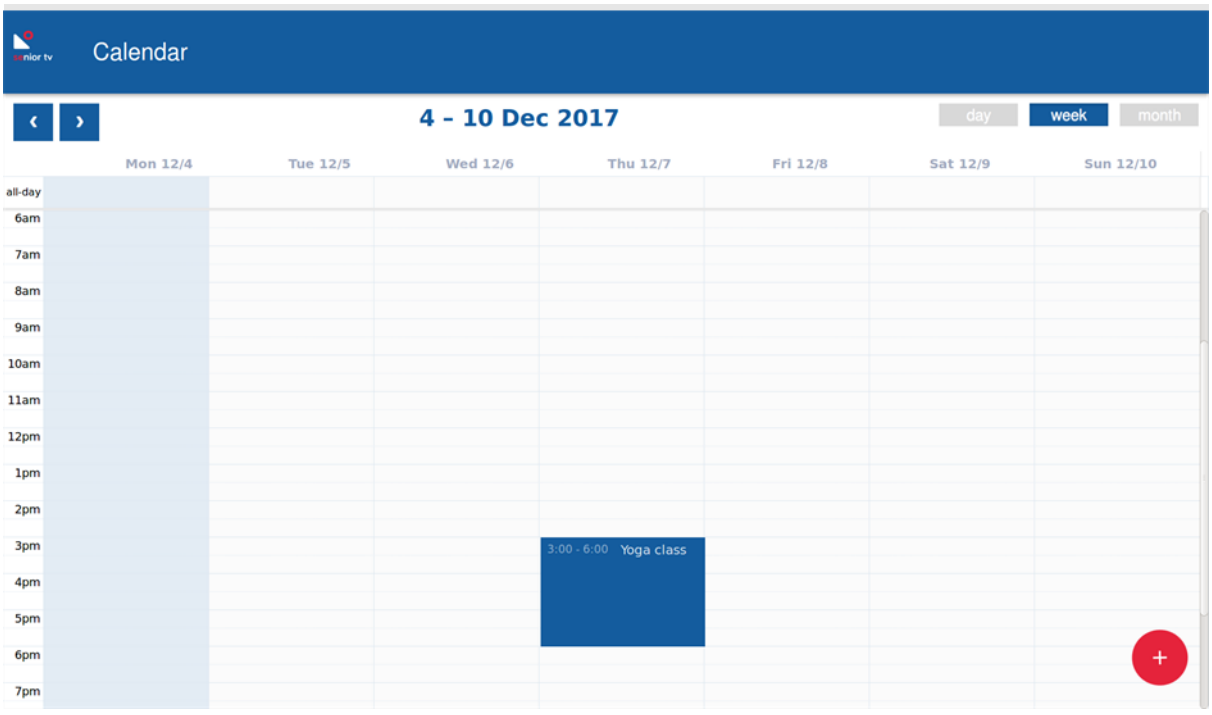


FIGURE 31. AGENDA WEB HOME SCREEN WEEKLY VIEW

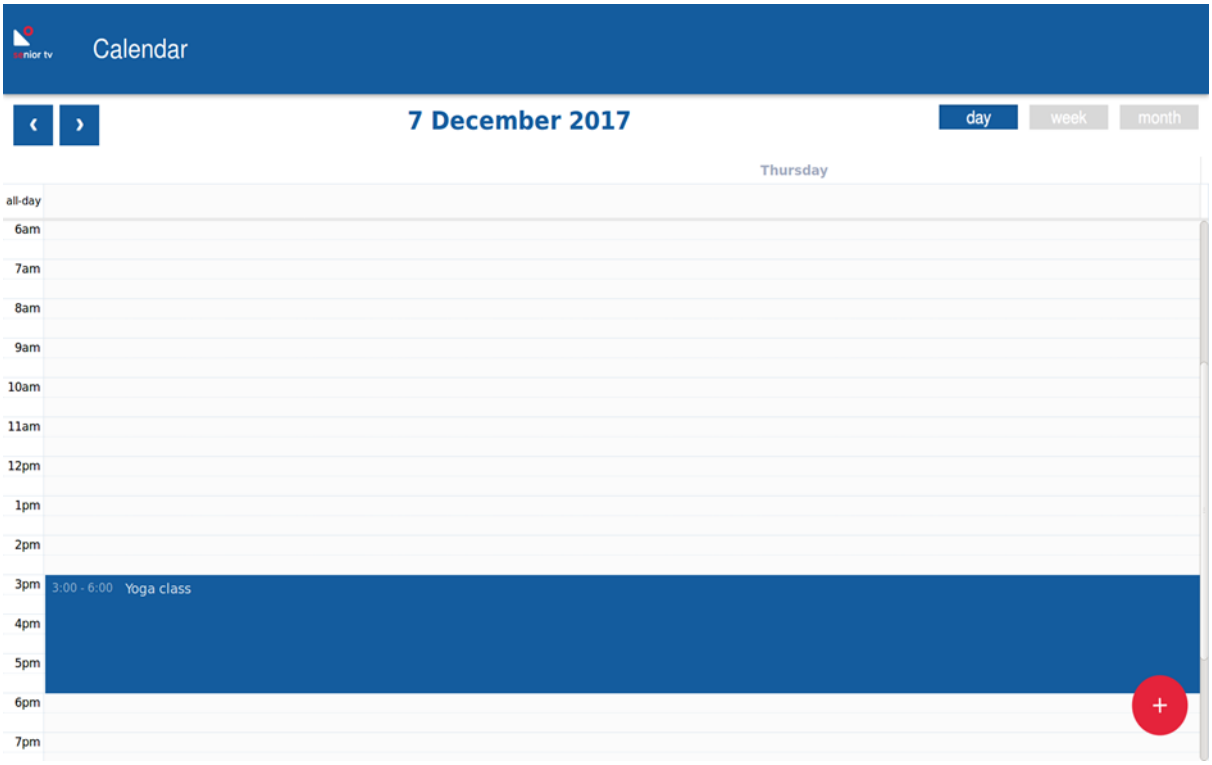


FIGURE 32. AGENDA WEB HOME SCREEN DAILY VIEW

2. Event Details Screen

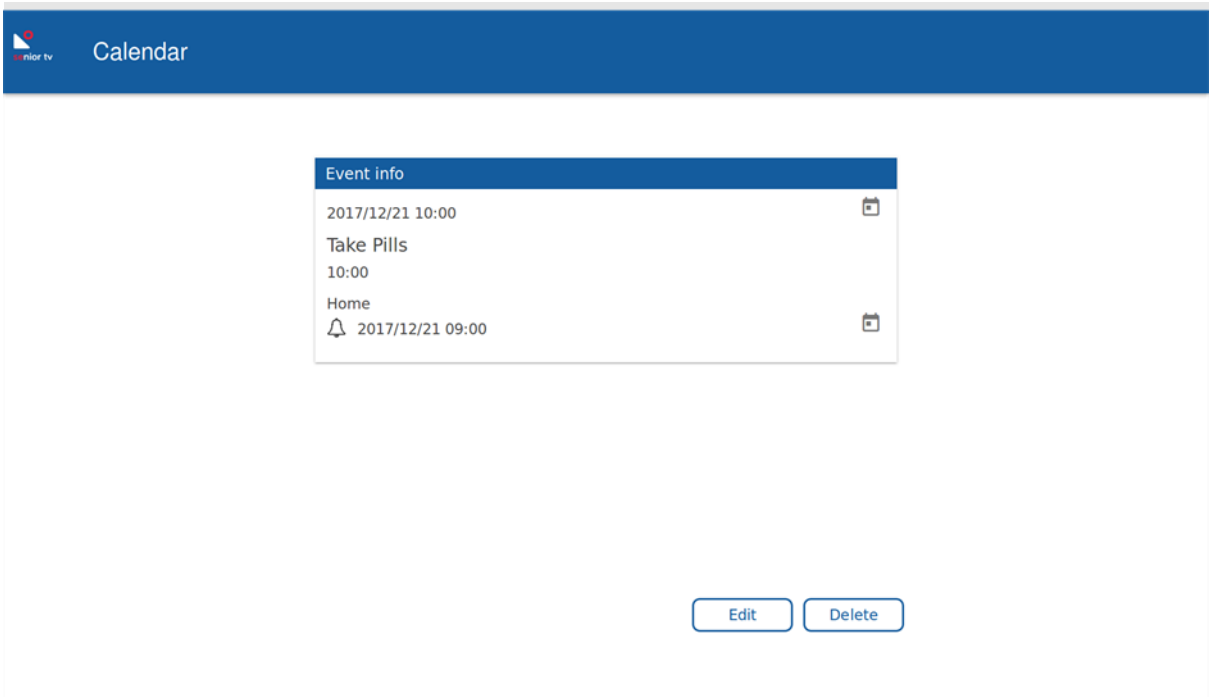


FIGURE 33. AGENDA WEB EVENT DETAILS SCREEN

3. Add Event Screen

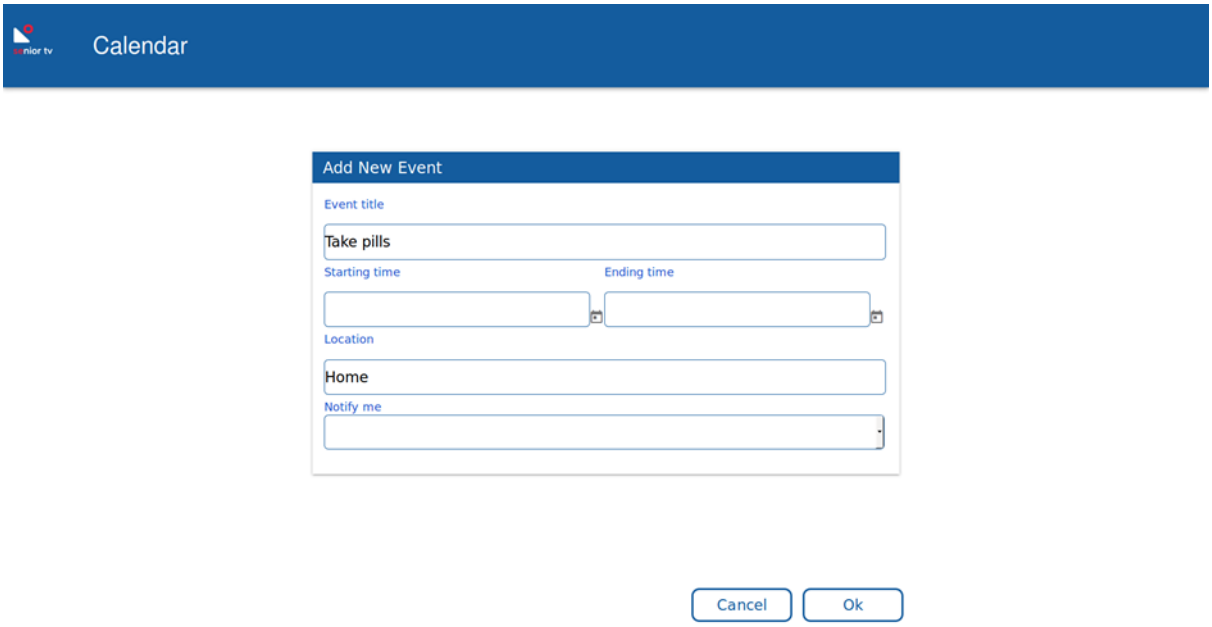


FIGURE 34. AGENDA WEB ADD EVENT SCREEN

4. Edit event Screen

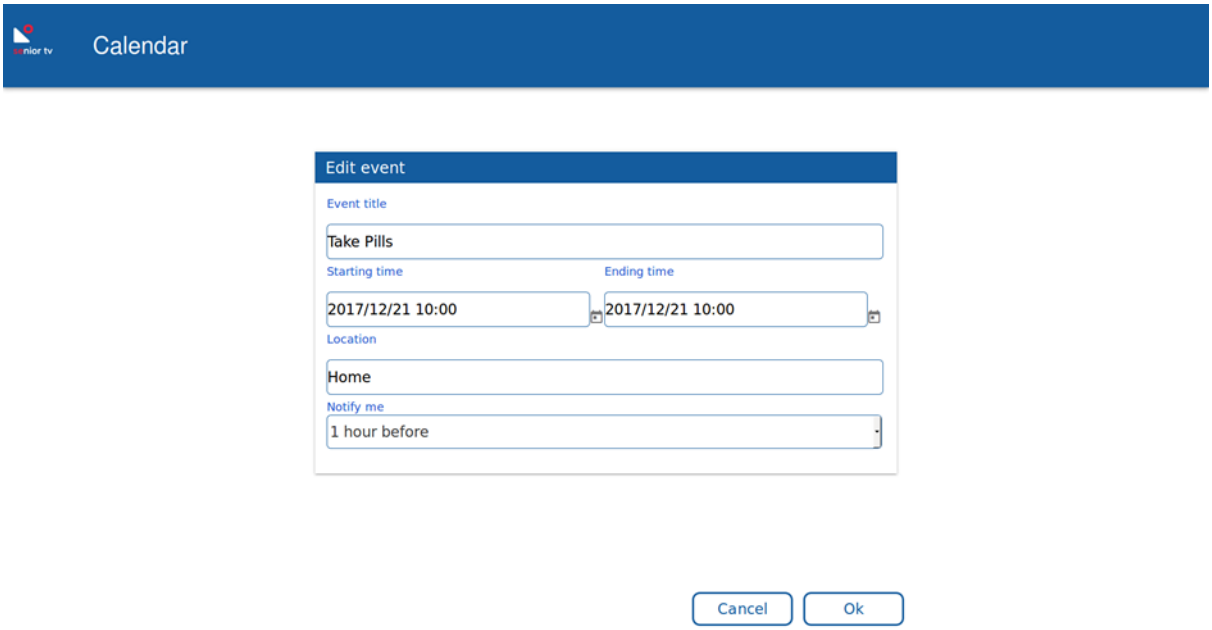


FIGURE 35. AGENDA WEB EDIT EVENT SCREEN

5.2.2 Events 2.0

This service is an upgrade of Events service developed for the first pilot. The initial version was mainly used to get the seniors' feedback about the interface usability the remote-control interaction and also the service itself.

The new version of this service is composed by two different modules: a TV and a Web app. The TV app is focussed on seniors and the Web app is oriented to secondary users.

5.2.2.1 Events TV

The scope of the TV app has not change from the initial version: use the TV screen to find and participate to events that seniors are interested in. The backend functionalities and the architecture of upgraded version have not changed from the initial one (see section 4.1.1). Although new developments were done in the version 2.0 in order to complete all mock-up functionalities shown in the initial version and to include the System Integration services. From the previous version the following improvements were achieved:

- Include the System Integration services.
- Integration with the Events Web app and Cloud.
- Integration with the Agenda service.
- Implementation of the new design guidelines.
- The app is completely controllable from only the D-pad, Home, OK and Back Buttons.
- A preference screen was implemented for personalizing the text size and the preferred categories.

For developing, the same hybrid technologies have been used than for initial version: Ionic 3, HTML, TypeScript, Cordova, and various open source third party plugins. The main screens of the TV interface are as follows:

1. The Category Selection Screen

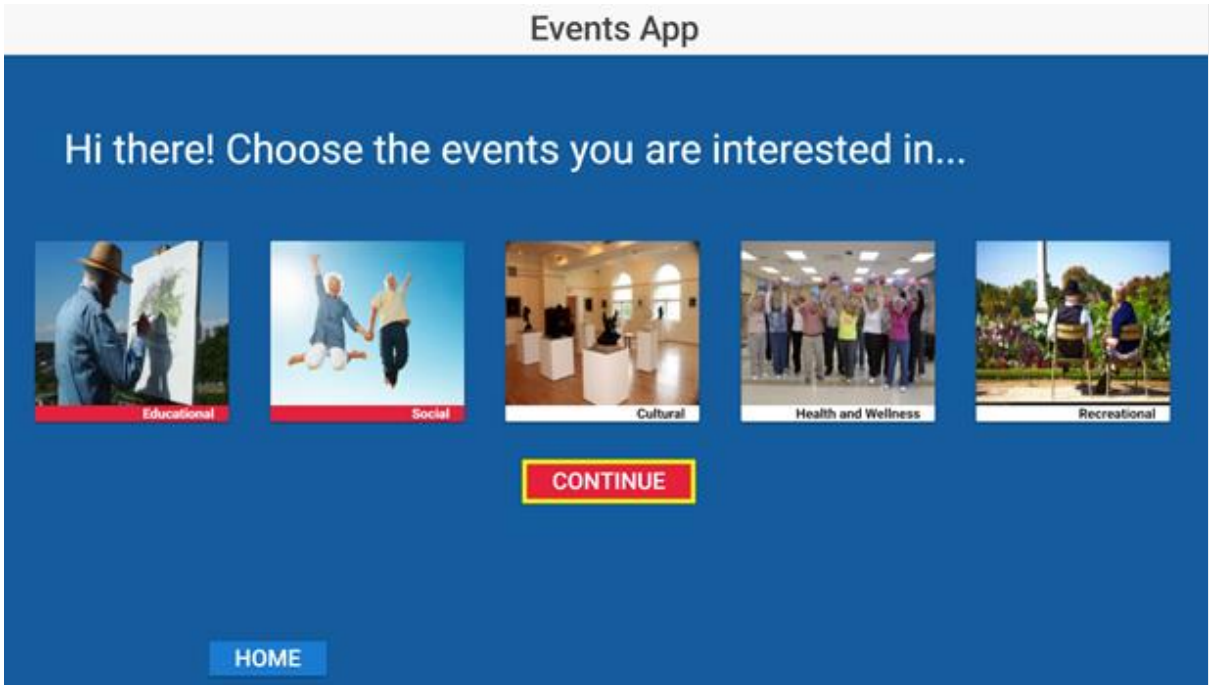


FIGURE 36. EVENTS 2.0 APP SELECTION SCREEN

2. The Events Preview Screen

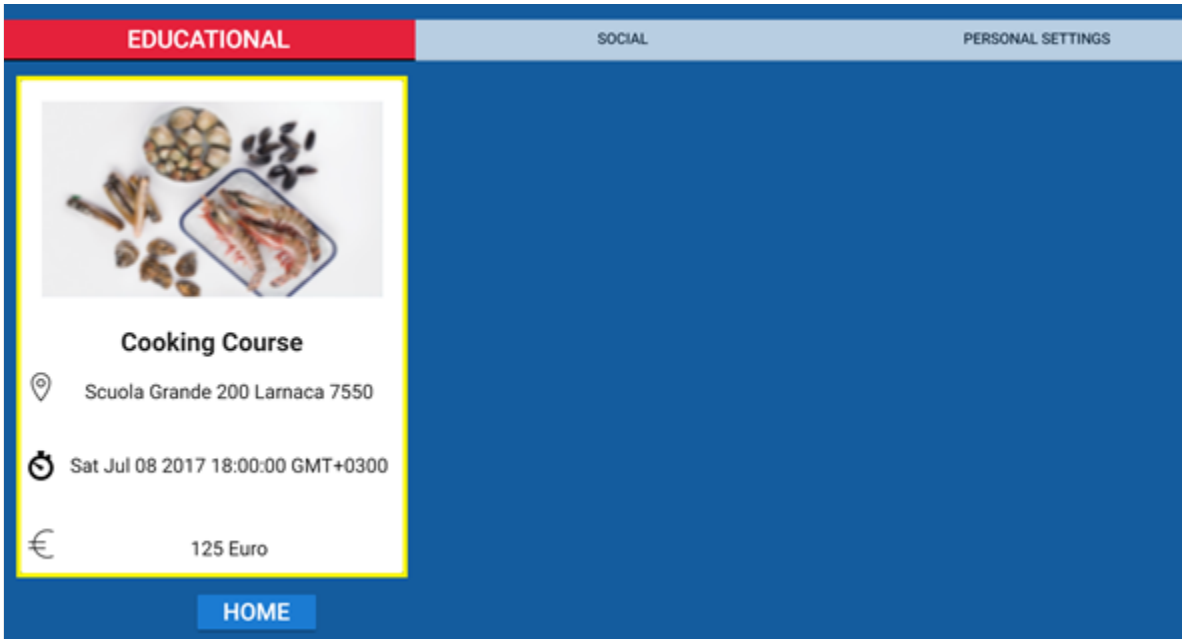


FIGURE 37. EVENTS 2.0 APP PREVIEW SCREEN

3. The Event View Screen

FIGURE 38. EVENTS 2.0 VIEW SCREEN

4. The Customization Screen

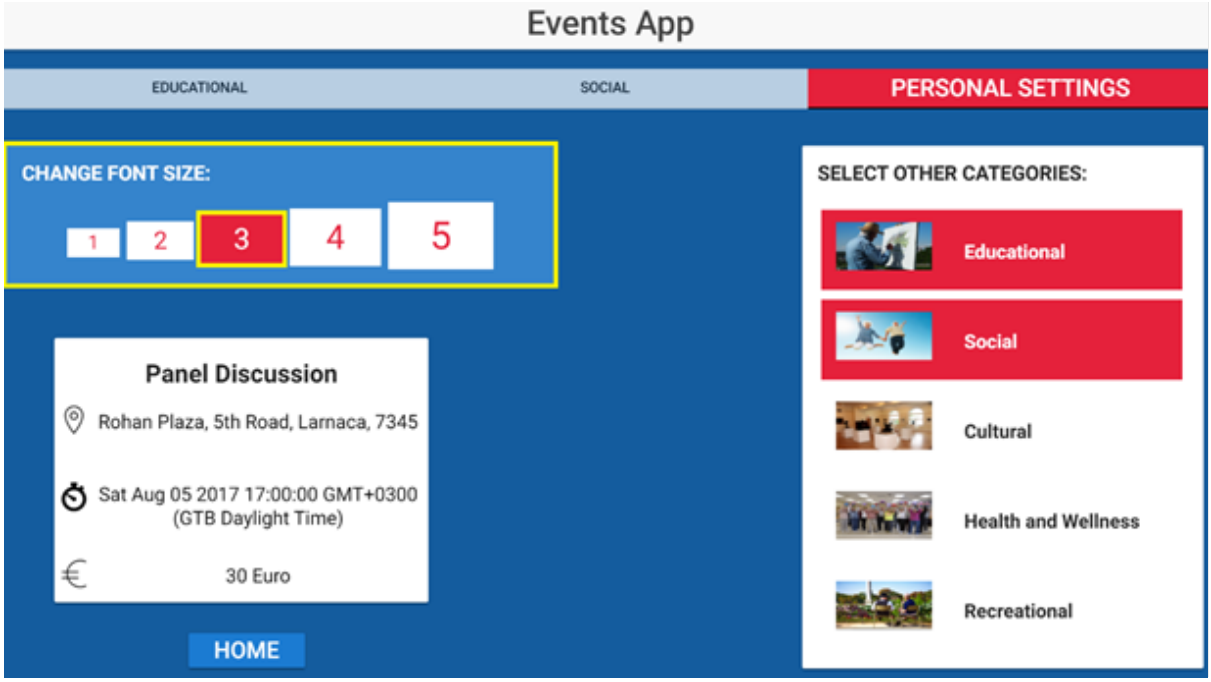


FIGURE 39. EVENTS 2.0 CUSTOMIZATION SCREEN

5.2.2.2 Events Web

The scope of this application is to enable different event providers for elderly to add events which will be shown through TV application. These events will be filtered when they are shown on the TV based on the preference for language for each SENIOR-TV system.

The Events Web app is accessible from this URL: http://5.2.183.39:30005/. The architecture of the app is shown below:

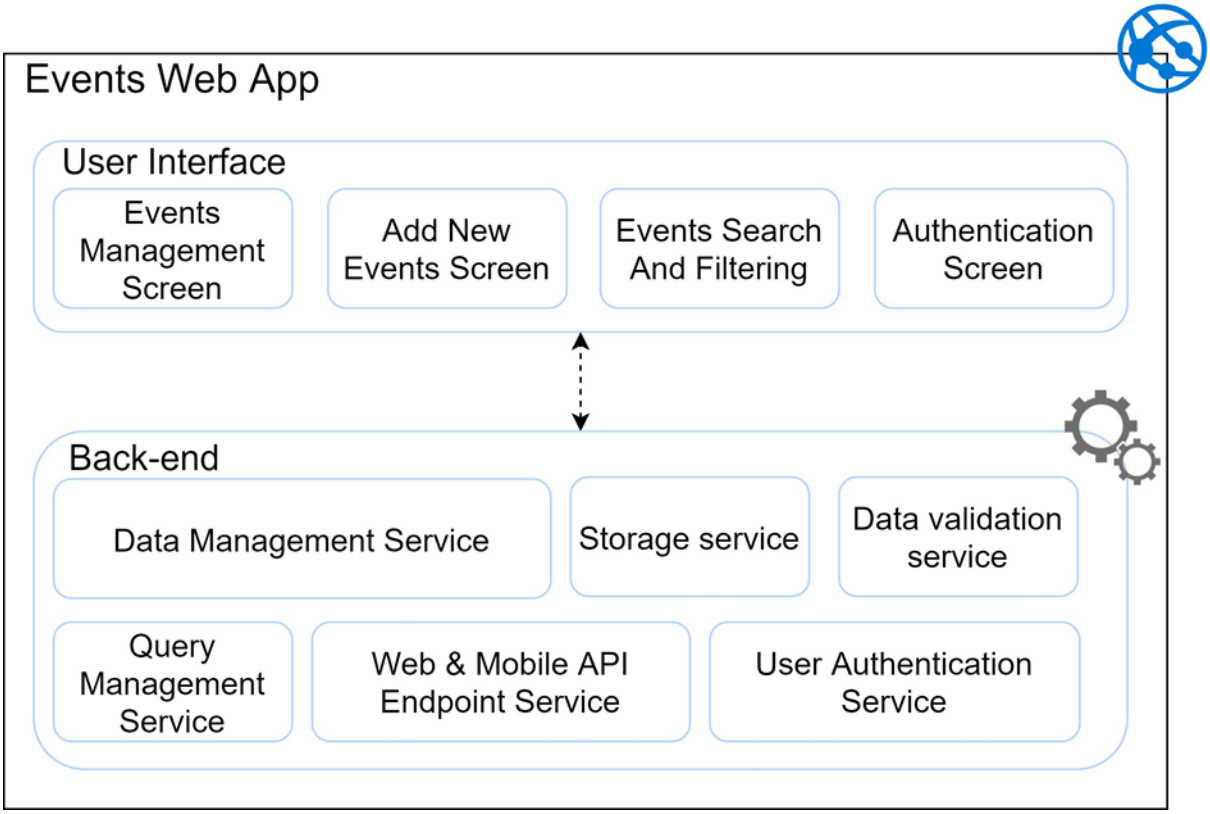


FIGURE 40. EVENTS 2.0 CUSTOMIZATION SCREEN

The components of the Events Web app are:

1. The Events Management Screen enables the editors of the events to view, add, remove and edit events.
    - Editors can add new events by clicking on the Add New Button. In order to add a new event some detailed information is required: Group (Social, Cultural, Recreational, etc.), Language, Type, Address, Date, Price, Details and, Image.
    - The Editors can search for events using a variety of comparison operators.
    - The Authentication Screen allows the user to log in to the application.
2. The Back-end of the Events service has the following components:

- Data Management Service: enables the management of data transferred between the APIs and the Storage Service.
- Storage Service: enables the persistence of Data in a SQL Database.
- Data Validation service: verifies that the queries and commands sent to the cloud server are authorized and in accordance to the business logic.
- Query management service: enables the dispatch of queries and commands from the API Controllers to the corresponding handlers.
- Web & Mobile API Endpoint Service: enables the Web App and Mobile app to exchange REST messages with the Cloud Server.
- User Authentication Service: enables the create, update and login/logout of users in the web services.

The technologies used for developing the Events Web app are: Angular4, HTML, TypeScript, .Net and various other open source and third-party libraries. The interface first version is shown below:

1. Add new event Screen

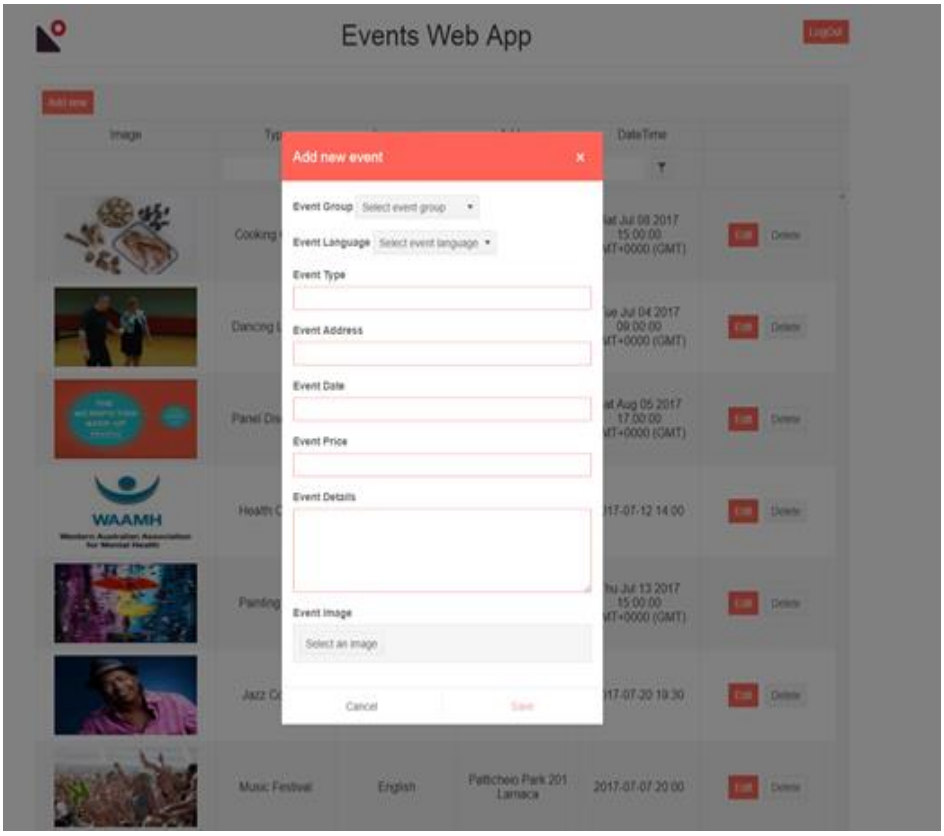


FIGURE 41. NEW EVENT SCREEN

2. List events Screen

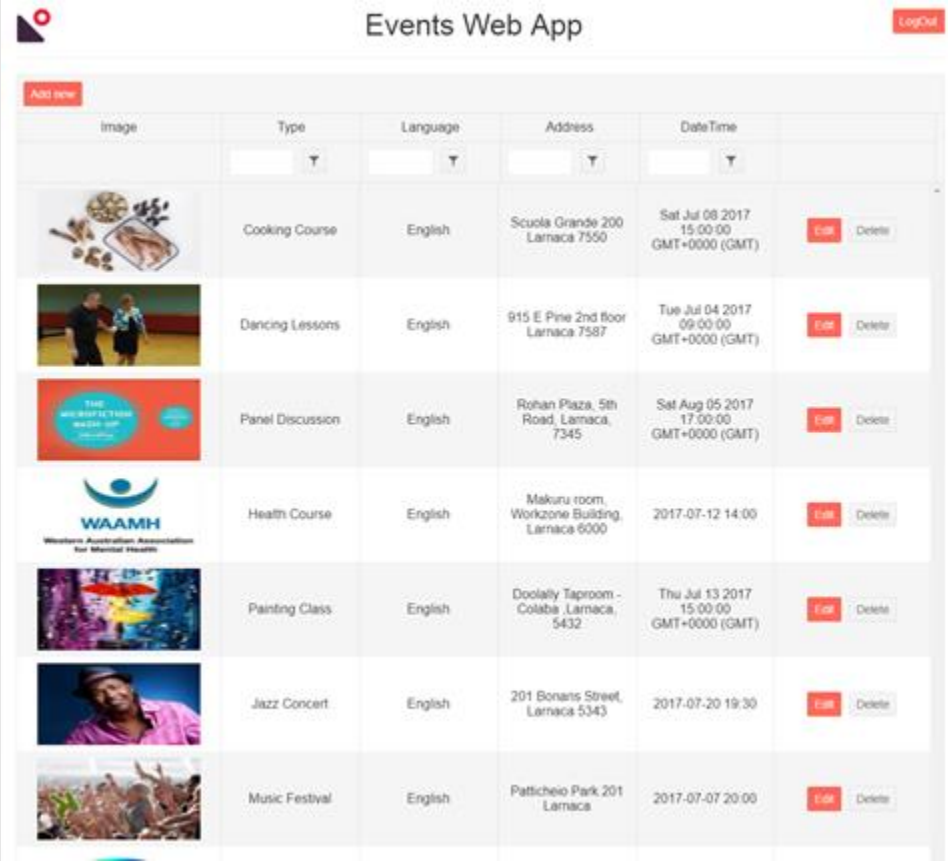


FIGURE 42. LISTS EVENTS SCREEN

5.2.3 News 2.0

This service is an upgrade of News service developed to first pilot. The initial version was only used to get the seniors feedback about the interface usability and the remote-control interaction. Although new developments were done in the version 2.0 in order to complete all mock-up functionalities shown in the initial version and to include the System Integration services.

From the previous version the following improvements were achieved:

- Integration with the Auth app and its services.
- Implementation of the new design guidelines.
- The app is completely controllable from only the D-pad, Home, OK and Back Buttons.
- A preference screen was implemented for personalizing the text size and the preferred categories.

For developing, the same hybrid technologies have been used than for initial version: Ionic 3, HTML, TypeScript, Cordova, and various open source third party plugins. Below are screenshots of the improved app:

1. News Central Screen

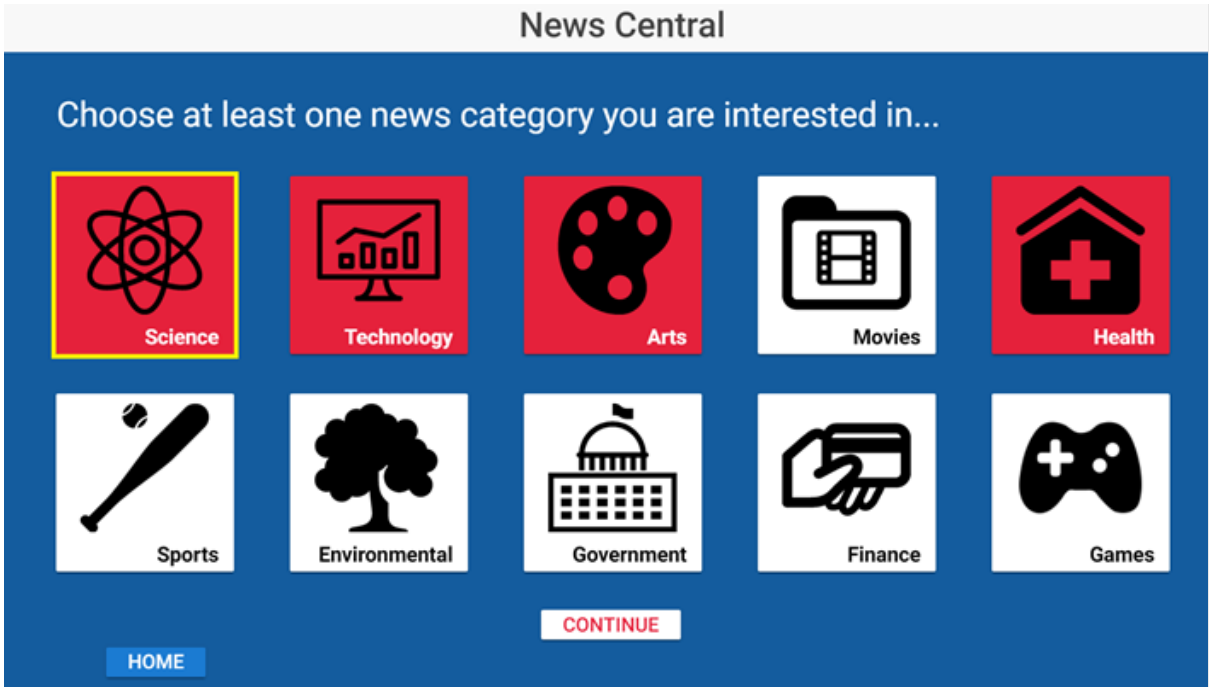


FIGURE 43. NEWS 2.0 CATEGORIES SELECTION SCREEN

2. Category List News Screen

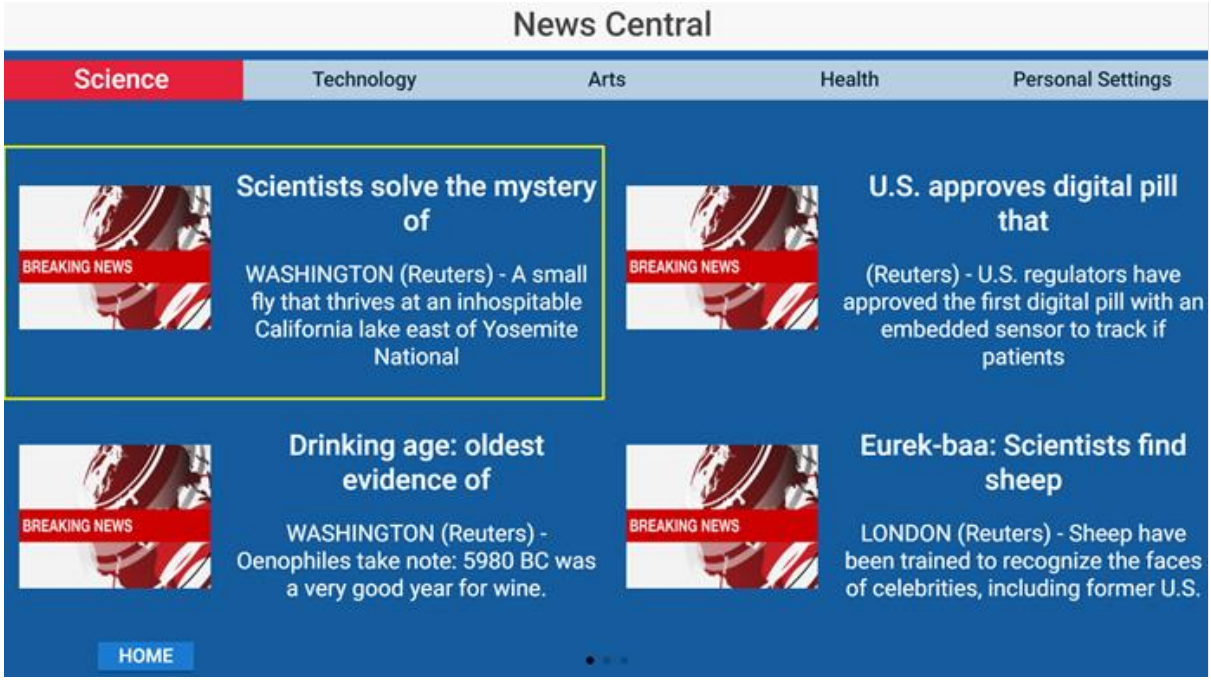


FIGURE 44. NEWS 2.0 CATEGORY LIST NEWS SCREEN

3. The Customization Screen

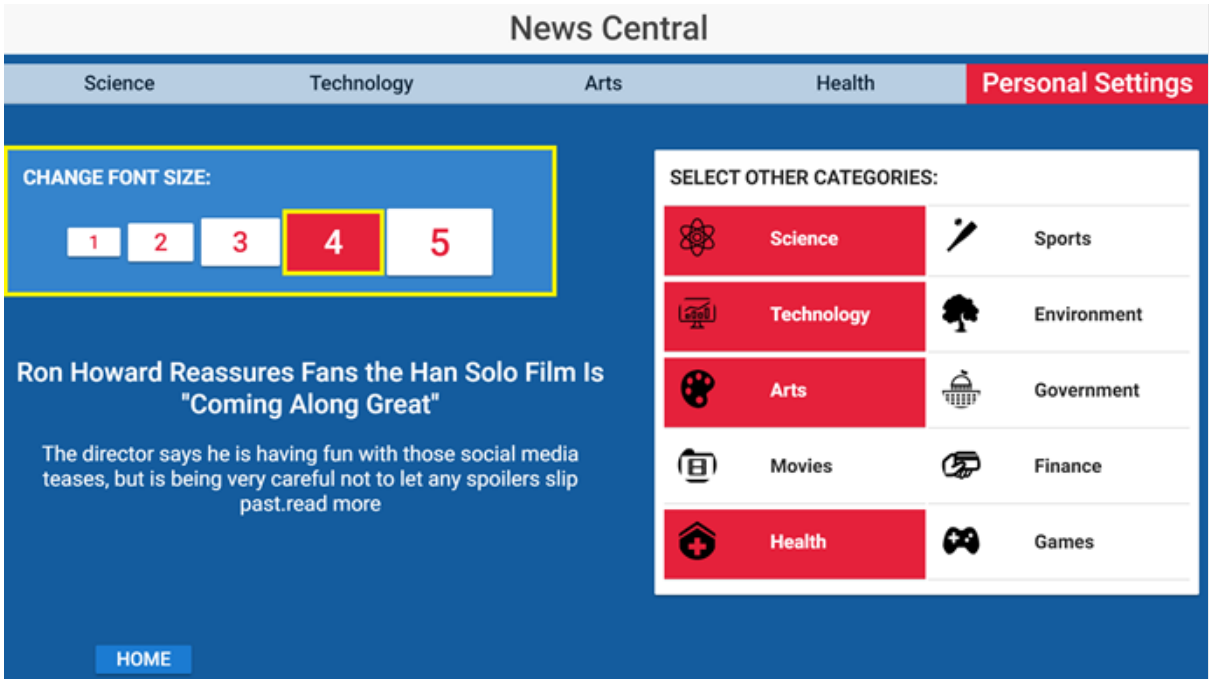


FIGURE 45. NEWS 2.0 CUSTOMIZATION SCREEN

5.2.4 Tracker 1.0

Keeping an active life brings a lot of benefits for seniors. Tracker service allows users to keep a record of their walks using a mobile phone and then, use the TV to see the walk metrics and results.

To encourage senior to do outdoor exercises the service includes a common gamified environment where friends can set up their own competitions: seniors not only can see their walks but also can see their "friends" results.

Besides, Tracker service provides mechanisms to reduce the seniors' sense of insecurity when they walk. The seniors' address can easily store in the mobile app in order to know the way home in case they felt lost or disoriented.

The Tracker service is composed by two different apps: a TV and a Mobile app. The Mobile app records the seniors' walks and the TV app shows the walks summarizes. The following chart shows the architecture of the service:

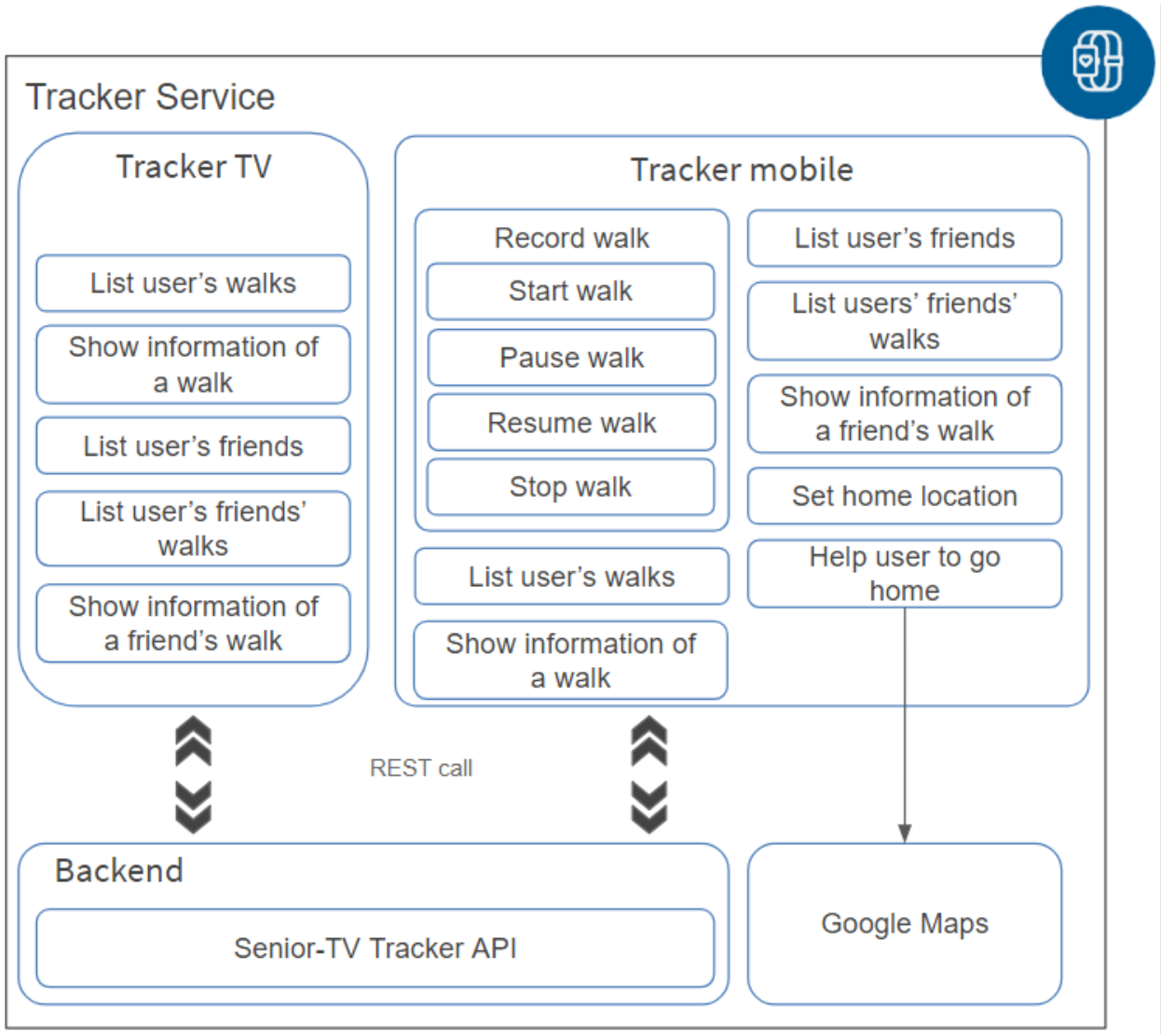


FIGURE 46. TRACKER ARCHITECTURE

Both, Mobile and TV apps have been implemented using Angular framework and Apache Cordova for TV app. The walk information is shown on a map in two apps. To paint this information the Leaflet library is used.

Leaflet[21] is a widely used open-source JavaScript library used to build interactive maps in applications. It supports most mobile and desktop platforms, supporting HTML5 and CSS3. Leaflet allows developers without a geographic information system to display tiled maps, with optional tiled overlays. It can load feature data from GeoJSON files, style it, and create interactive layers, such as markers with popups when clicked.

[21] Leaflet official site: https://leafletjs.com/

The cloud backend has been developed using Java EE technology. The two apps use HTTP request to communicate with the backend to obtain the walks' data. The cloud backend is composed by two services:

- SENIOR-TV Tracker API: allows manage the favorite locations (add, remove or update the database information).
- Google Maps[22] is an external service used to calculate the route from the seniors' location to their home.

The next paragraphs describe the TV and Web applications architecture, their functionalities and the main interface screens.

5.2.4.1 Tracker TV

The scope of the TV app is to enable elderly users to see their walk metrics from the TV. The Tracker TV main functionalities are listed below:

1. List user's walks: allows users to see all their walks.
2. See user's friends' walks: users can see the walk details made by their friends. Before to access the walks' data, it is necessary to select one of their friends.
3. Show a walk information: allows users to see the details of a walk. In particular, users can see the following pieces of information:
    - Route on a map.
    - Walk Time.
    - Walk Total time.
    - Walked distance.
    - Average speed.

The following charts show the Tracker TV app architecture and some interface screens:

[22] Google Maps Platform: https://cloud.google.com/maps-platform/

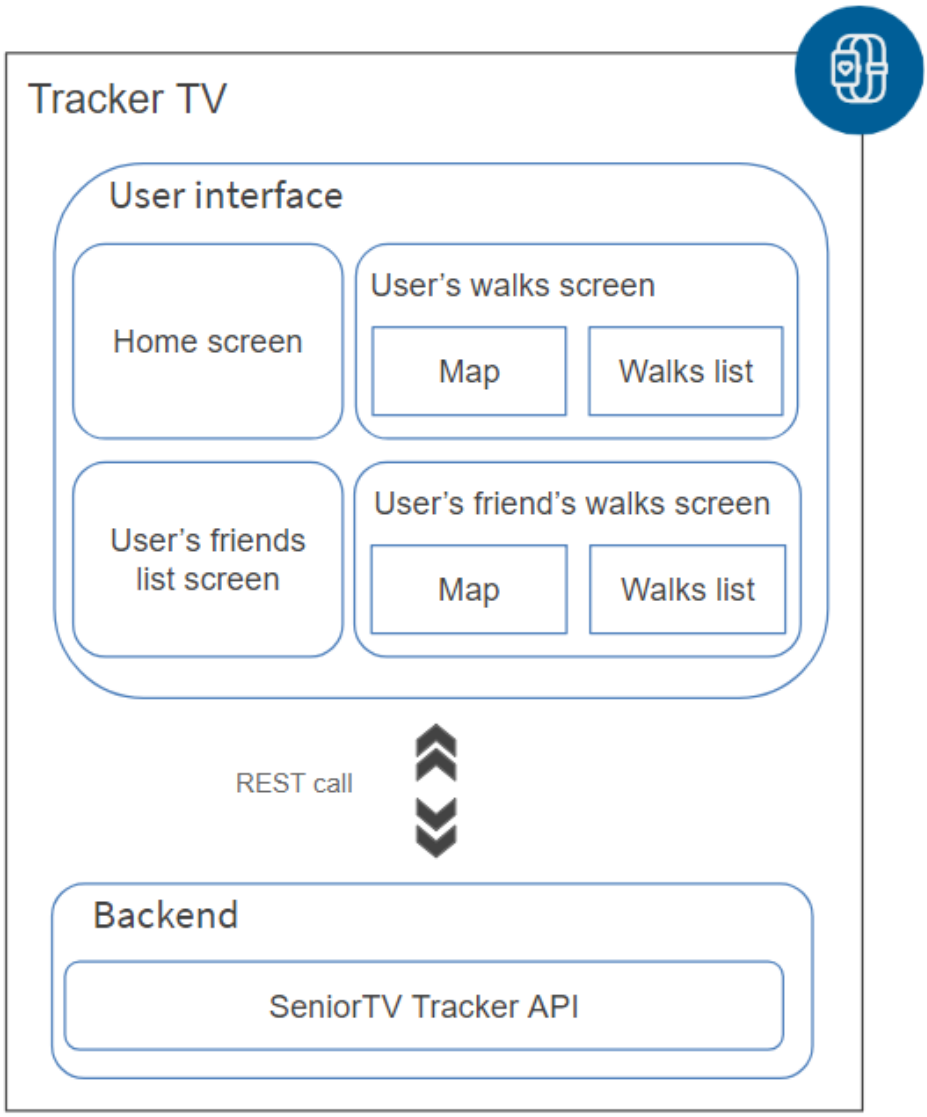


FIGURE 47. TRACKER TV ARCHITECTURE

1. Home screen

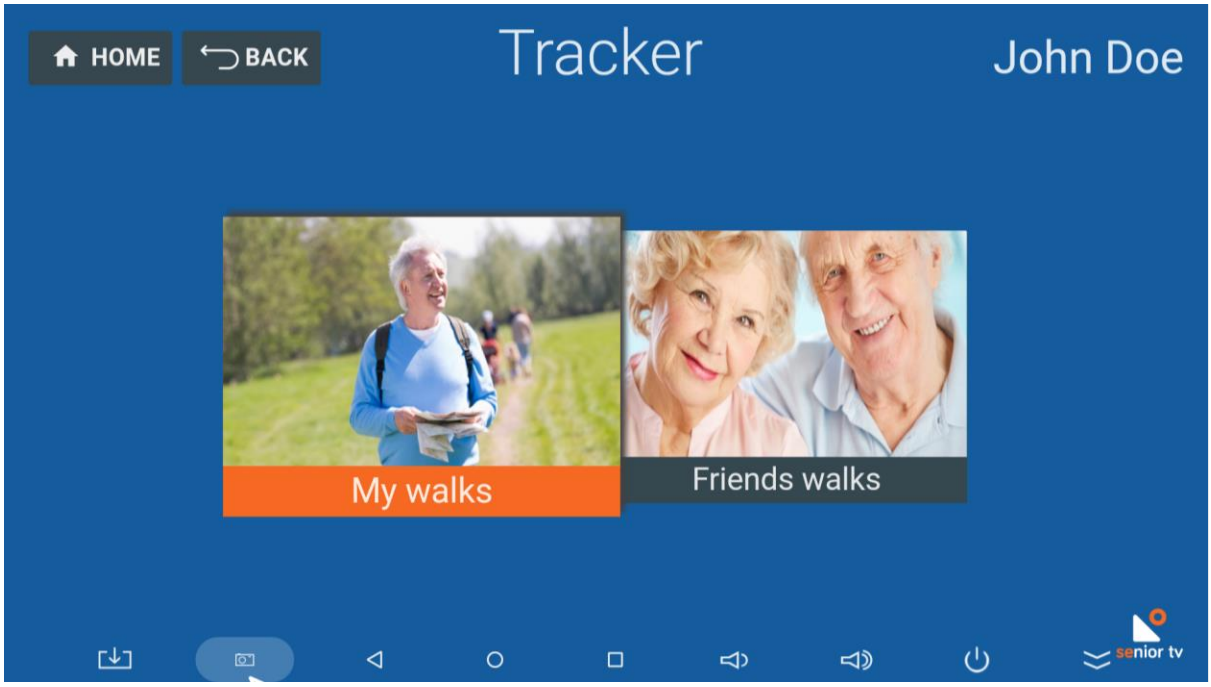


FIGURE 48. TRACKER TV HOME SCREEN

2. Walk list screen

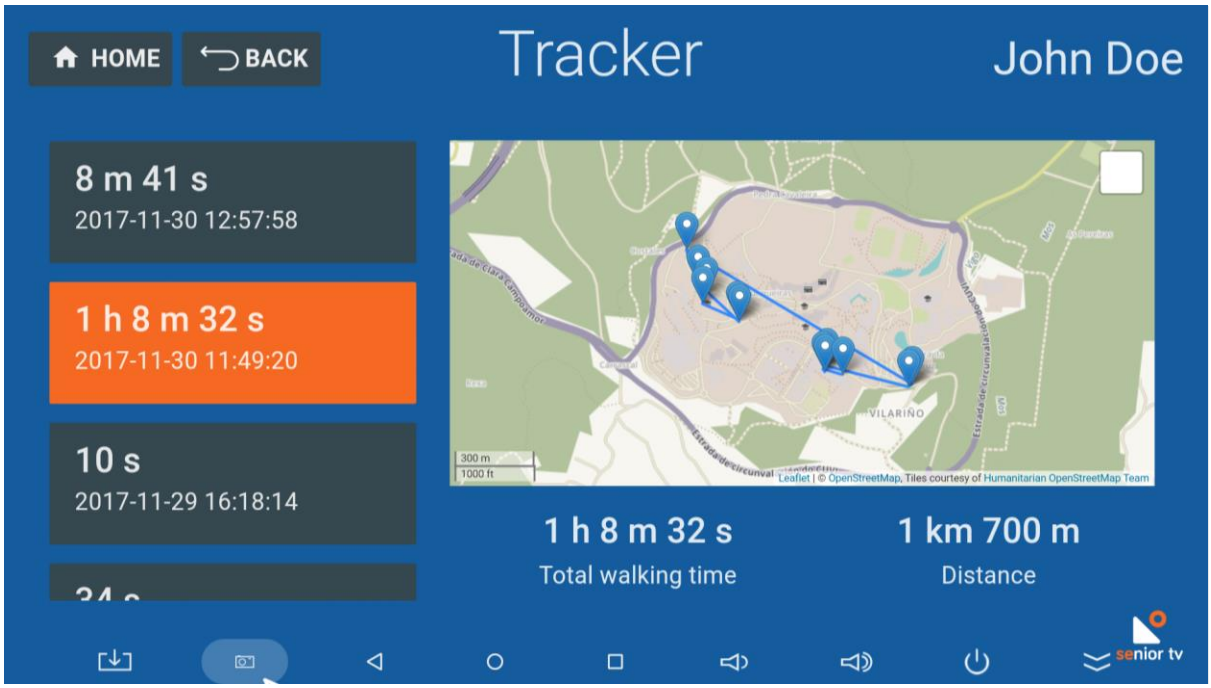


FIGURE 49. TRACKER TV USER'S WALKS SCREEN

3. Friend list screen

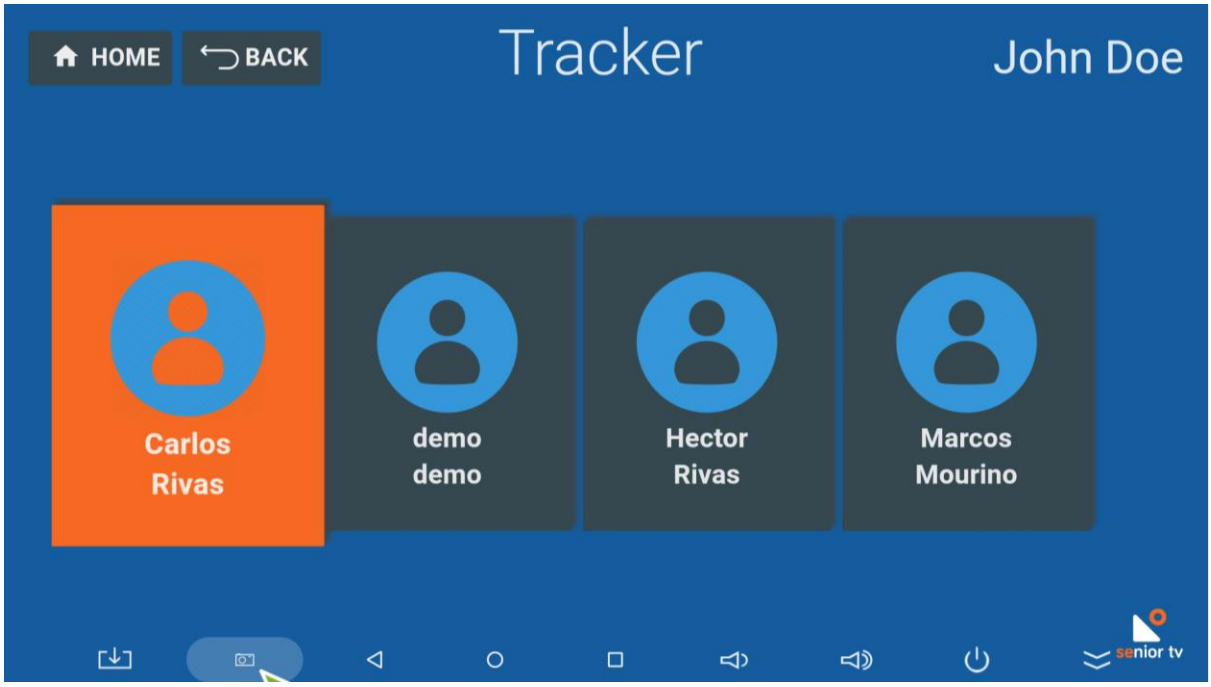


FIGURE 50. TRACKER TV FRIENDS LIST SCREEN

5.2.4.2 Tracker Mobile

The mobile app includes all TV functionalities and adds some more regarding walks recording and "my home" route. These are the specific Tracker Mobile functionalities:

1. Record walk: collects users' walk information, such as: starting time of a walk, GPS positions, walked distance, average speed, ending time and, total walked time.
   - *Start walk*: Once the walk starts, the application gets the current GPS position, and sends to the Tracker Cloud the starting time and the initial position of the walk. Then, every 15 seconds, the application gets the current GPS position and sends it to the Cloud. Besides, the application calculates the current distance walked and average speed of the walk.
   - *Pause walk*: allows users to stop the data collection. While the walk is paused, the application does not send GPS information to the Cloud.
   - *Resume walk*: allows restore the data collection.
   - *Stop walk*: finalises the data collection. The application sends to Tracker Cloud the ending time, the total walked distance, the average speed and, the total walked time.
2. Set home location: allows users to establish the location of their homes. Every user can only have one *Home position*. Therefore, when the user sets a home location the previous information will be deleted.

3. Help user to go home: shows the route and indications from senior's location to their home. This functionality is only available after seniors' *Home location* set.

The following charts show the Tracker TV app architecture and some interface screens:

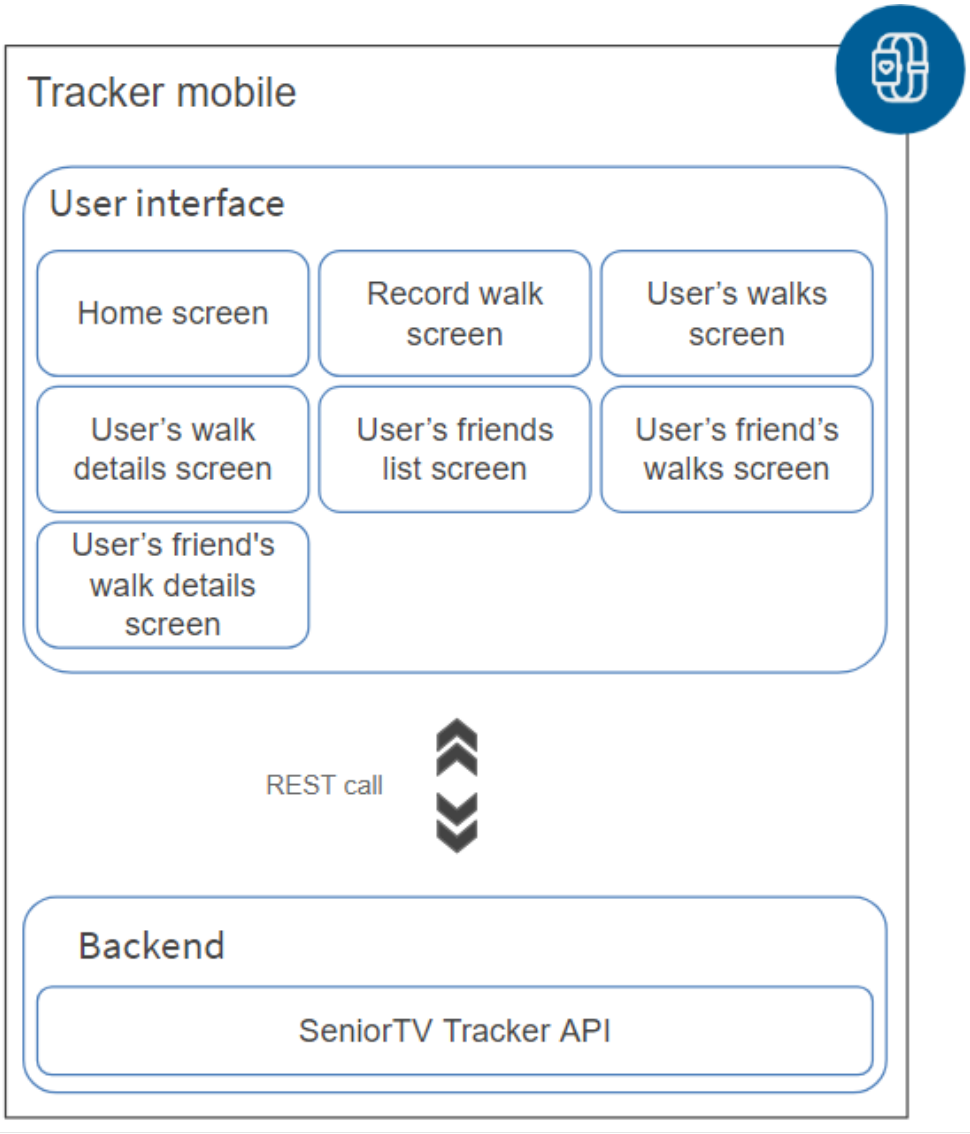


FIGURE 51. TRACKER TV ARCHITECTURE

1. Home screen

FIGURE 52. TRACKER MOBILE HOME SCREEN

2. Record walk screen

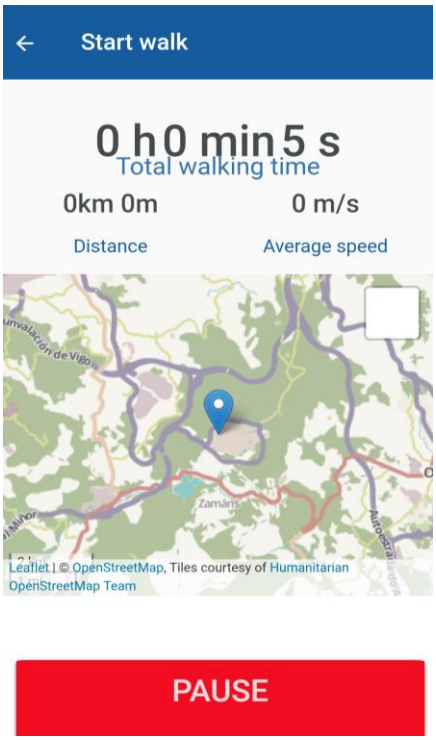


FIGURE 53. TRACKER MOBILE RECORD WALK SCREEN

3. Walk list screen

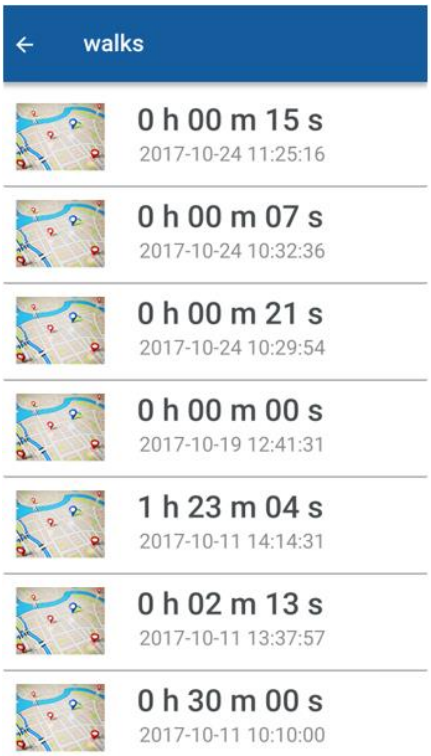


FIGURE 54. TRACKER MOBILE WALK LIST SCREEN

4. Walk details screen

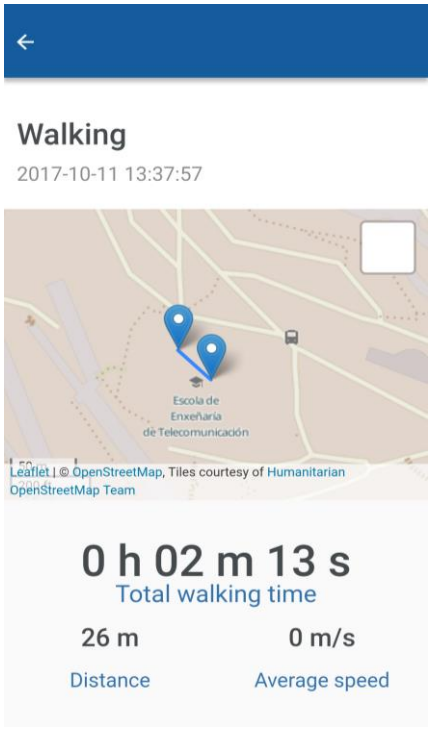


FIGURE 55. TRACKER MOBILE WALK DETAILS SCREEN

5. Friend list screen

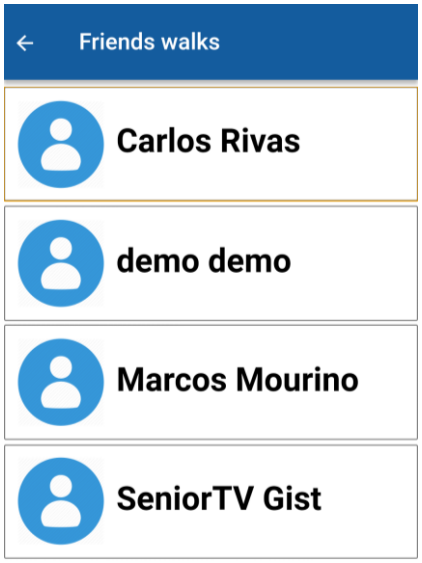


FIGURE 56. TRACKER MOBILE FRIEND LIST SCREEN

6. Friend walks list screen

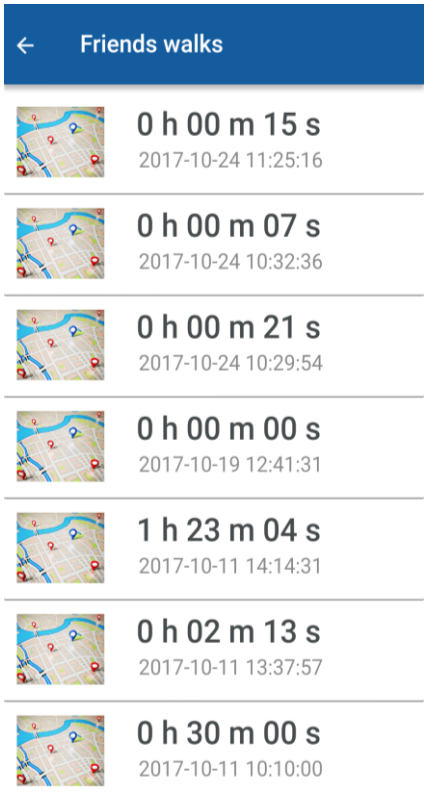


FIGURE 57. TRACKER MOBILE FRIEND WALKS LIST SCREEN

5.2.5 Virtual Center 1.0

There are many reasons why an older person should stay at home reducing their contact with the outside and disrupting their daily routine. The Virtual Center service allows seniors to keep contact with their daily care center from their home through the TV. Elderly associations can stream the activities which take place in their facilities (courses, gym maintenance, etc.) and seniors will be able to follow these activities without need to move until their center.

This service also may be used as a complement of daily care center activities. For example, doing morning exercises at the center and evening exercises at home.

The Virtual Center service is composed by two elements: the Streamer and the TV app. The TV app is focussed on seniors and the Streamer is oriented to daily care centres. The following chart shows the architecture of the service:

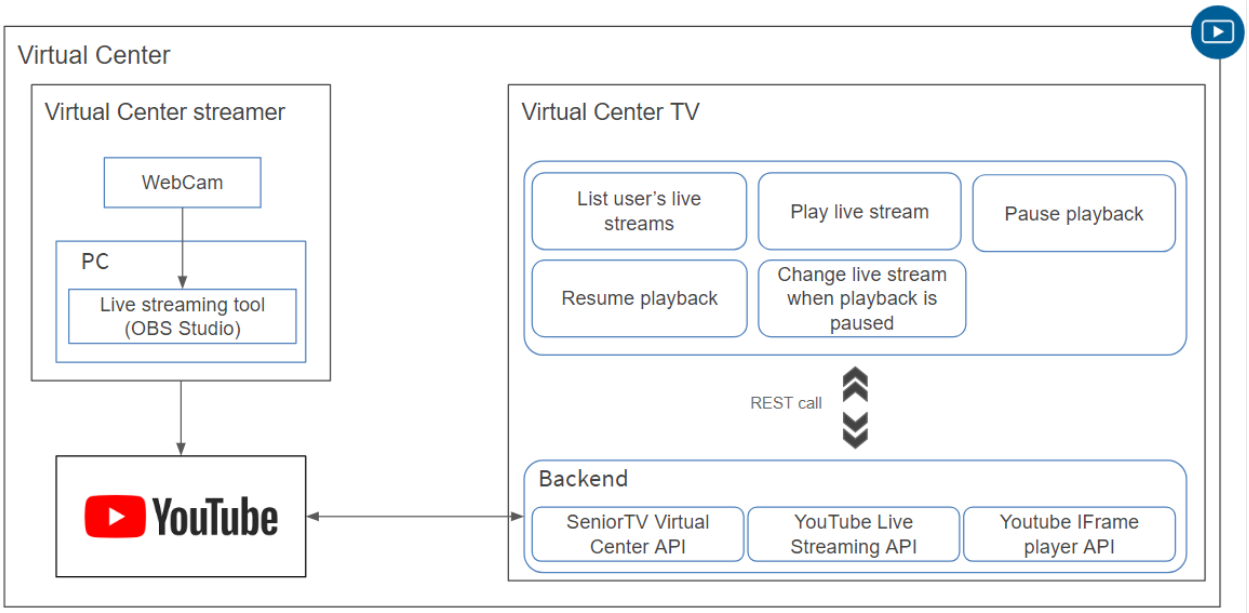


FIGURE 58. VIRTUAL CENTER GENERAL ARCHITECTURE

5.2.5.1 Virtual Center Streamer

The Streamer is responsible to stream the elderly centre activities. It is composed by a webcam, a microphone and a computer with a live streaming tool installed. In particularly, in this project the *Open Broadcaster Software*[23] (OBS) is used because it is an open source tool for Windows, macOS and Linux.

The OBSS tool allows to stream audio and video using through a channel YouTube[24]. YouTube is a video-sharing service that allows users to upload, view, rate, share, add to favourites, report, comment on videos,

[23] Open Broadcaster Software Studio official site: https://obsproject.com/
[24] YouTube official site: https://www.youtube.com/

and subscribe to other users. Among the features provided by YouTube, it allows to create live streams in order users can real-time broadcast events. Also, YouTube has an open Live Streaming API[25] which permits update and manage live events easily.

5.2.5.2 Virtual Center TV

This application is responsible for playing on the TV the Virtual Center streams. Seniors can navigate through the channels and selected one of them to watch the current online streaming.

The next chart shows the Weather TV architecture. The cloud backend is composed by three services:

- SENIOR-TV Virtual Center API: allows manage the seniors channels.
- YouTube Live Streaming API: lets create, update, and manage live events on YouTube. It allows schedule events and relate them to video streams, which represent the broadcast content.
- Youtube IFrame player API: permits to embed a YouTube video player on a website and control the player using JavaScript. Queue videos for playback (play, pause, or stop videos), adjust the player volume or retrieve information about the video being played.

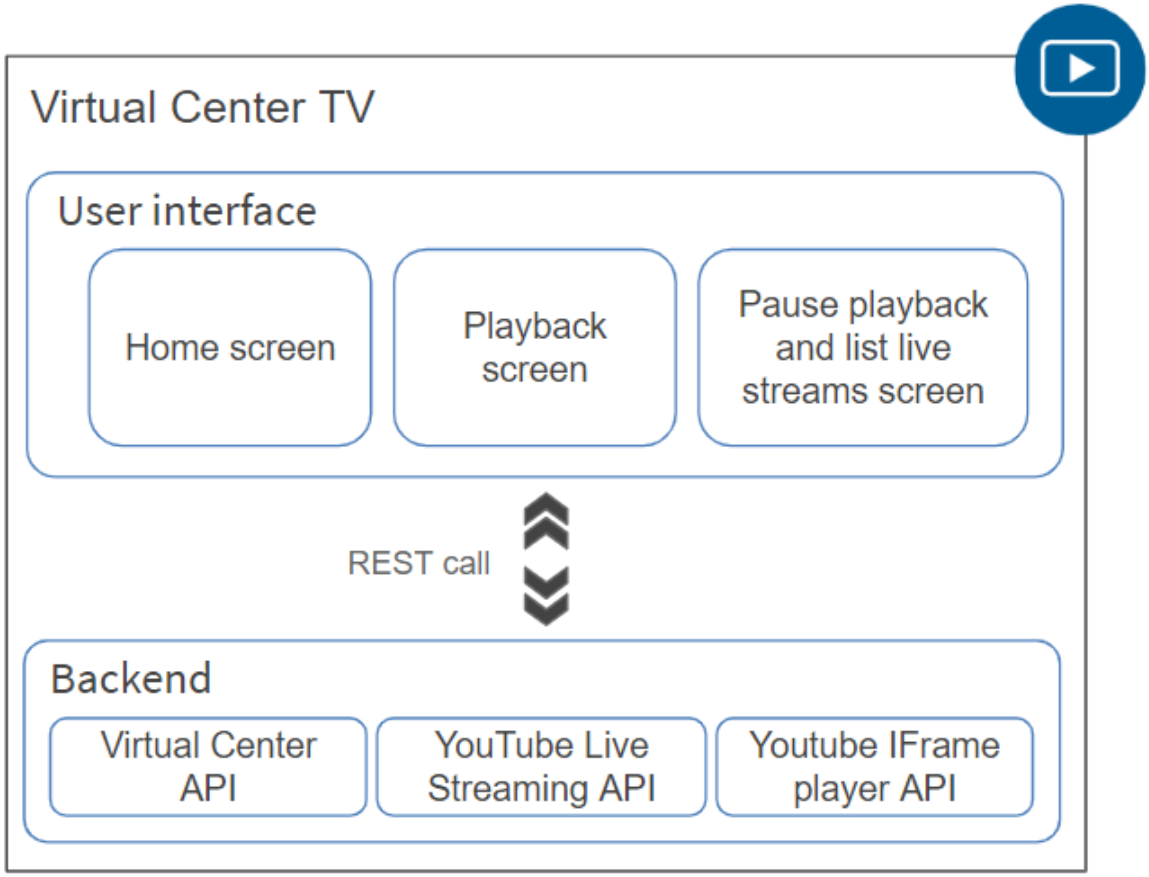


FIGURE 59. VIRTUAL CENTER TV ARCHITECTURE

[25] YouTube Live Streaming API site: https://developers.google.com/youtube/v3/live/getting-started

The Virtual Center TV app has been implemented using Angular and Apache Cordova frameworks. These are the app main functionalities:

1. Play live stream: allows the user to play a specific live stream.
2. Pause playback: allows the user to pause the playback of the current live stream using the Youtube IFrame Player API.
3. Resume playback: allows the user to resume the playback of the current live stream using the Youtube IFrame Player API.
4. Change to another live stream: when the playback is paused, seniors can change to another live stream channel from the video streaming screen.

The following charts show the main interface screens of the TV app:

1. Home screen

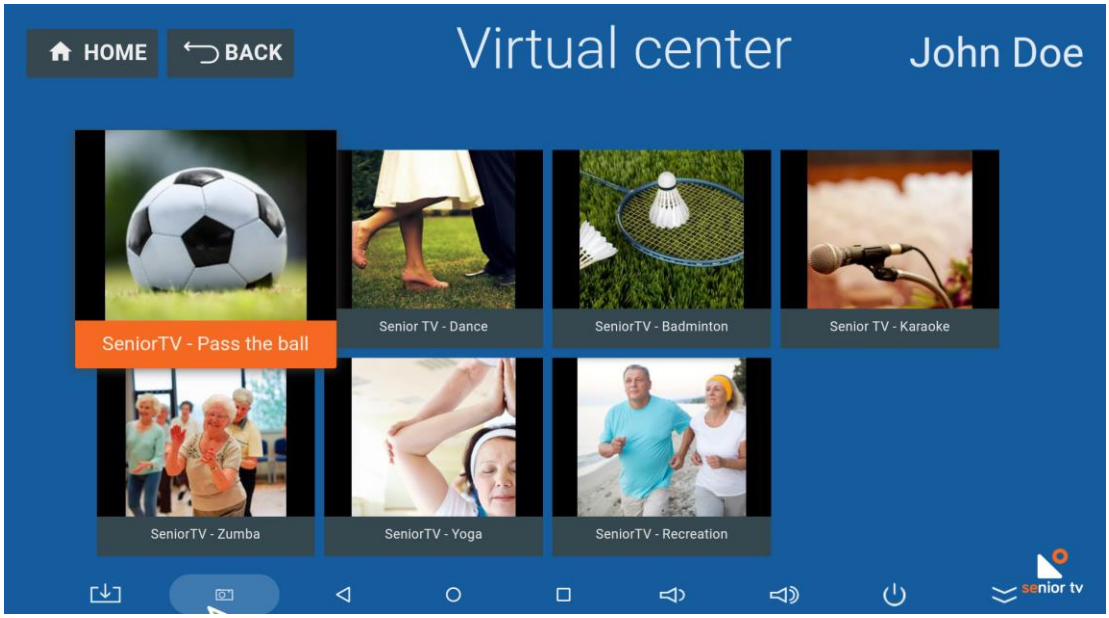


FIGURE 60. VIRTUAL CENTER HOME SCREEN

2. Playback screen

FIGURE 61. VIRTUAL CENTER PLAY BACK SCREEN

3. Pause playback and list channels screen

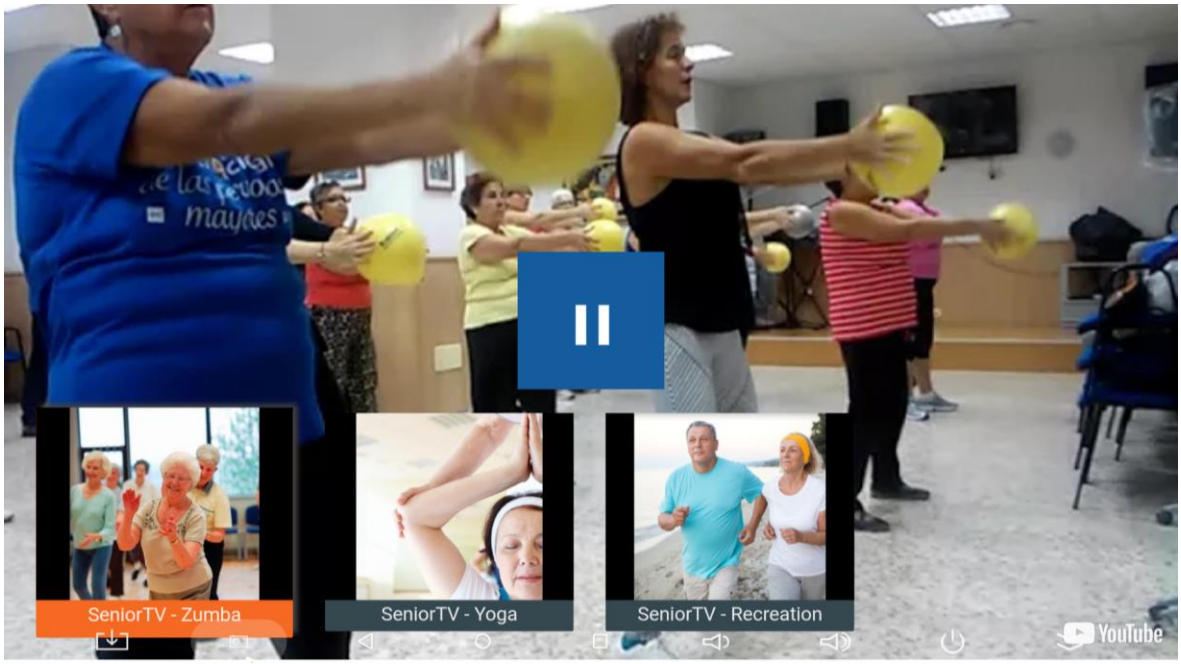


FIGURE 62. VIRTUAL CENTER PAUSE PLAYBACK AND LIST CHANNELS SCREEN

5.2.6 Weather 2.0

This service is an upgrade of Weather service developed for the first pilot. The initial version was mainly used to get the seniors' feedback about the interface usability the remote-control interaction and also the service itself.

The new version of this service is composed by two different modules: a TV and a Web app. The TV app is focussed on seniors and the Web app is oriented to secondary users. The following chart shows the service general architecture:

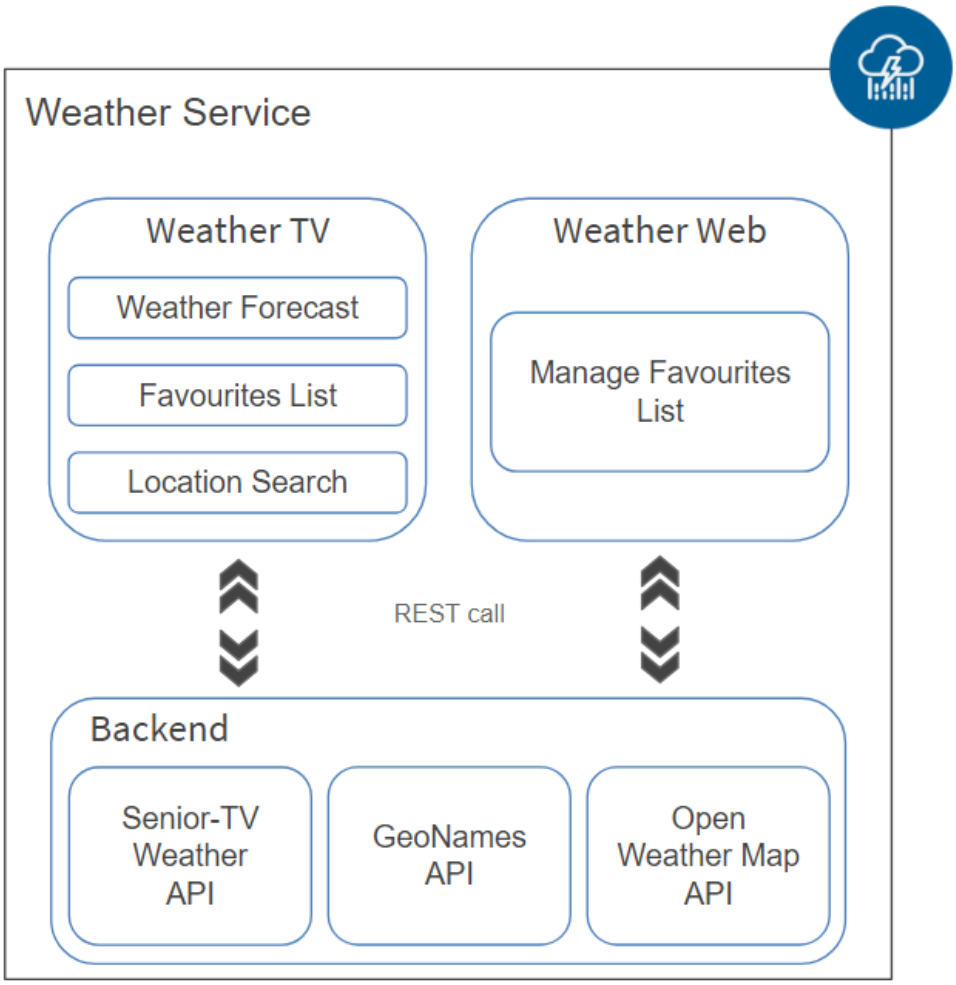


FIGURE 63. WEATHER GENERAL ARCHITECTURE

Both, Weather TV and Web, have been developed using Angular and Apache Cordova frameworks. The cloud backend has been developed using Java EE technology. The two apps use HTTP requests to communicate with the backend to obtain the seniors' favorite locations. The cloud backend is composed by three services:

- SENIOR-TV Weather API: allows manage the favorite locations (add, remove or update the database information).
- GeoNames[26] API: is used to translate the location names to their geographical latitude and longitude coordinates to avoid potential spelling errors. A location name could be different spellings depending on language. This issue can cause errors in the forecasts given by OpenWeatherMap API because it cannot identify the selected location.
- OpenWeatherMap[27] API: obtains the selected location weather forecast using the geographic coordinates obtained previously from GeoNames API.

The next paragraphs include the TV and Web applications architecture and the main interface screens description.

5.2.6.1 Weather TV

The scope of the TV app has not changed from the initial version: to use the TV screen to obtain the weather forecast of senior's location or other locations. Although the version 2.0 incorporates some new functionalities:

1. Search other locations forecast: users have to type the location name using the virtual keyboard.
2. Manage favourite locations: users can mark a location as favourite to not retype the location name any more.

The next chart shows the Weather TV architecture:

[26] GeoNames official site: http://www.geonames.org/
[27] OpenWeatherMap official site: https://openweathermap.org/

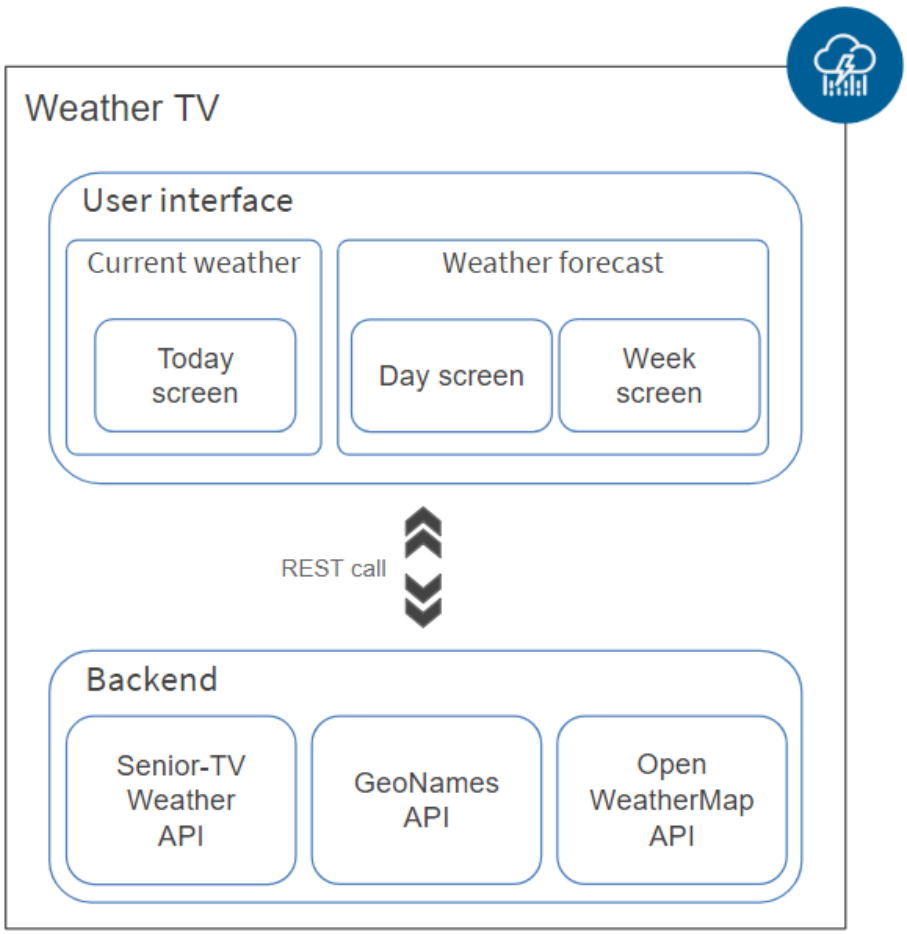


FIGURE 64. WEATHER TV GENERAL ARCHITECTURE

A new interface design was created for the Weather 2.0 to adapt the color palette and font sizes to the suggestions made by elderly during first pilot. The main screens are included following:

1. Daily forecast screen

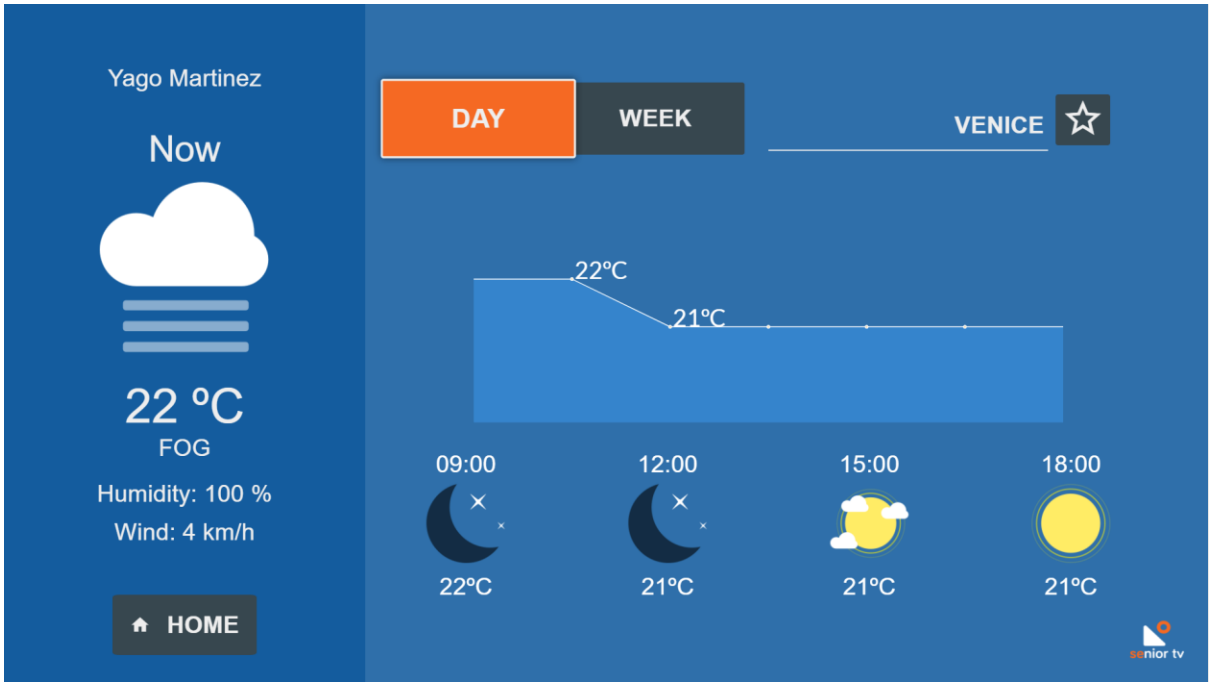


FIGURE 65. WEATHER TV DAILY FORECAST

2. Weekly forecast screen

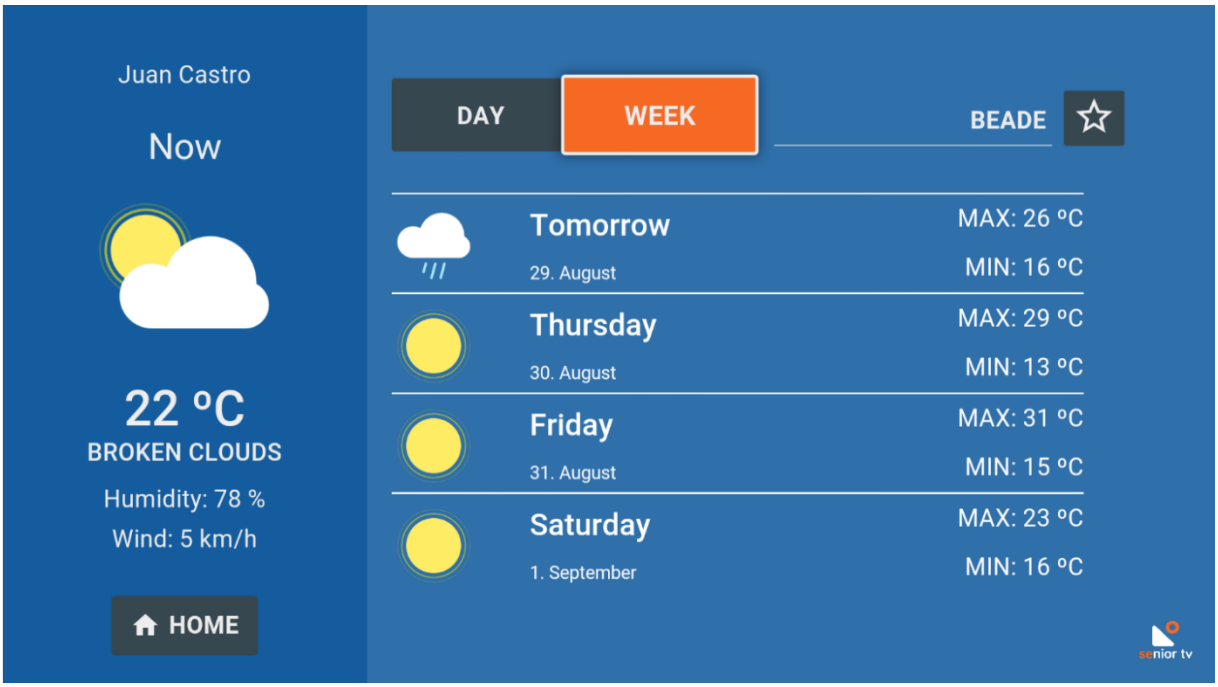


FIGURE 66. WEATHER TV WEEKLY FORECAST

5.2.6.2 Weather Web

The Weather Web app allows managing the seniors' favourite locations. This application is focussed to secondary users (caregivers or relatives) but primary users can also access to the app. These are the app main functionalities:

1. Add a favourite location to a particular user.
2. Remove elements from to a user favourite list.

The Weather Web app is accessible from this URL: https://www.imatia.com/seniortv-weather-web/. The next charts show the app architecture and some interface screens:

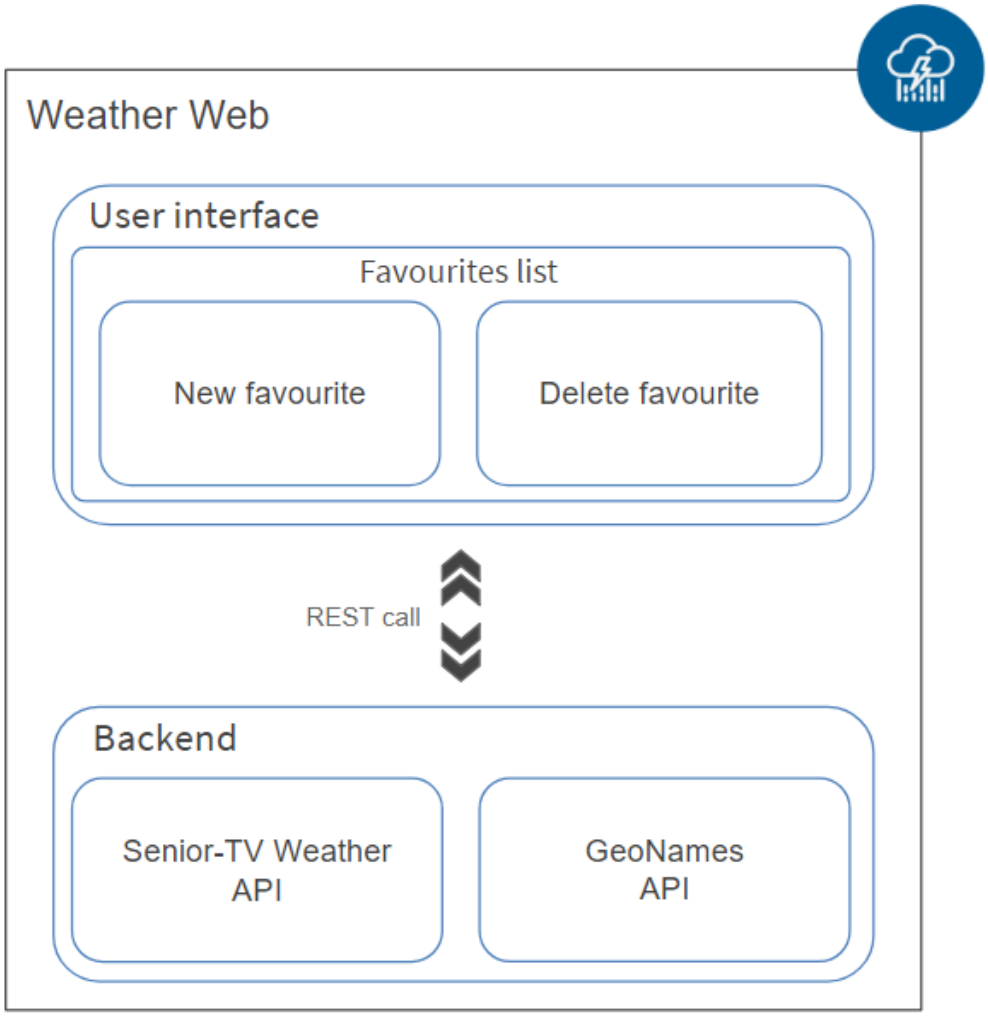


FIGURE 67. WEATHER WEB GENERAL ARCHITECTURE

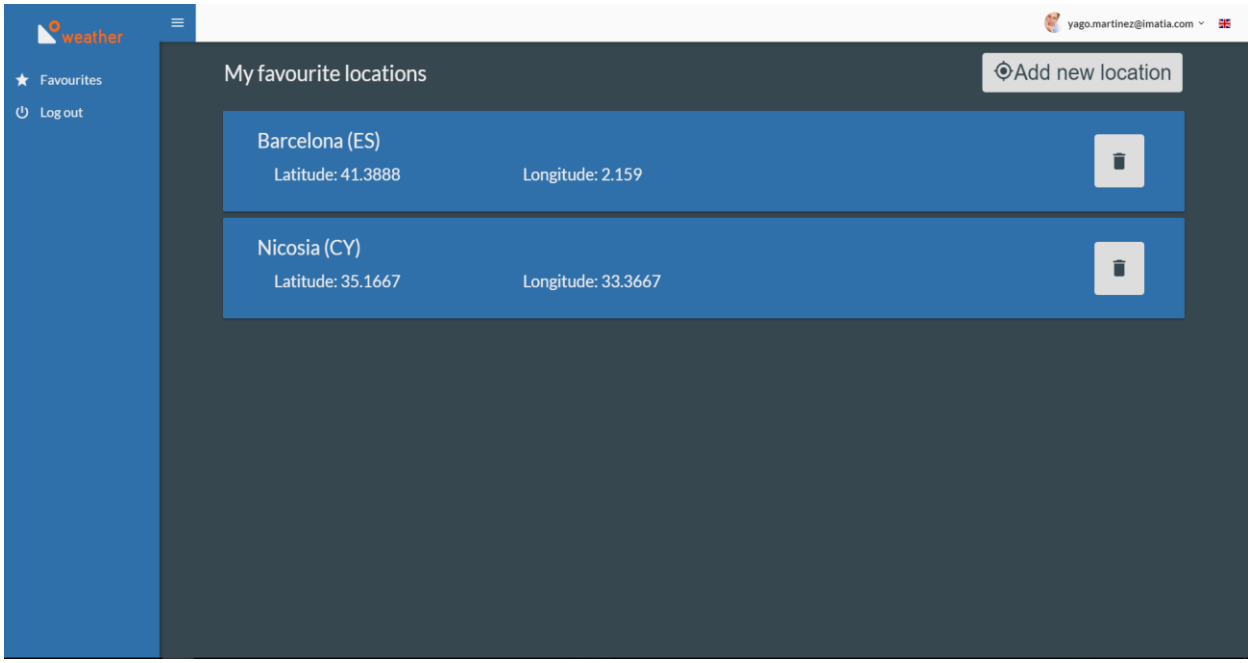


FIGURE 68. WEATHER WEB INTERFACE

5.2.7 Wikipedia 1.0

Wikipedia is a global, free and multilingual online encyclopaedia, with the mission of allowing anyone to read, create or edit articles. It currently houses more than 46 million articles distributed in 277 editions, being the largest and most popular general reference work on the Internet.

Access to Wikipedia allows seniors to access a large amount of information increasing their contact with external world. The Wikipedia service is a simplified senior-oriented Wikipedia client for TV. The service is designed to interact with the well-known web service in an easy and simple way using the TV. Users can save their usual searches or mark the articles as favourite to read later. Besides, they can use different Wikipedia versions depending upon the language (e.g. the English version is much larger than other minority languages).

The Wikipedia service consists only of the Wikipedia TV app which has been developed using JavaScript programming language and the Apache Cordova framework. The application fetches the data via HTTP requests to the public MediaWiki action API[28]. This API provides convenient access to wiki features, data, and metadata over HTTP requests.

The following chart shows the service architecture which comprises the functionalities described below.

[28] MediaWiki action API official side: https://www.mediawiki.org/wiki/API:Main_page/en

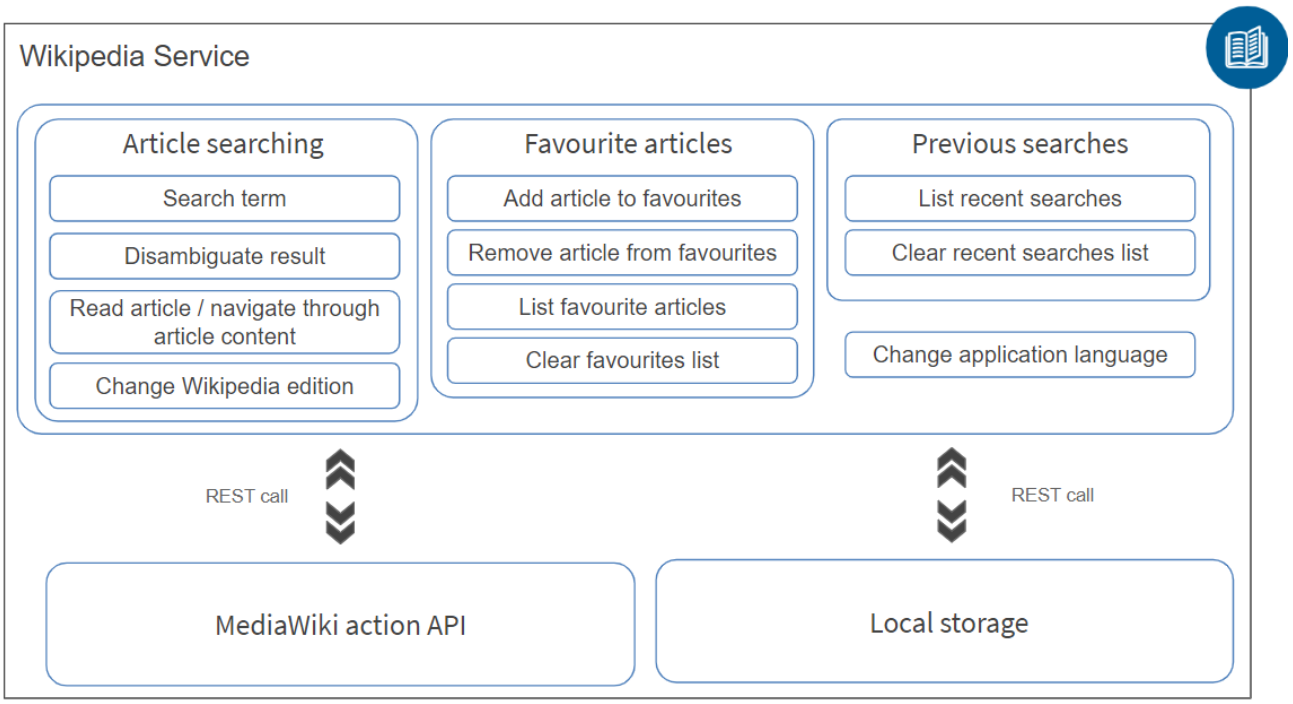


FIGURE 69. WIKIPEDIA GENERAL ARCHITECTURE

The service functionalities are organized in three groups: article searching, favorite articles, previous searches.

1. Search an article: allows to look for a specific article in a particular Wikipedia version.
2. Read article or Navigate through article content: permits to read the selected Wikipedia article content and to navigate through its different sections.
3. Manage favorite articles: seniors can bookmark articles in order to access them later.
4. Manage recent articles: the TV app stores the latest read articles references automatically. Users can access to the recently viewed articles list in order to repeat the searches without typing search terms specifically.
5. Change Wikipedia edition: allows users to perform requests on different Wikipedia editions. The current available editions are English, Spanish, Greek, Romanian, and Slovenian.
6. Change application language: permits to translate the display texts to several languages: English, Spanish, Greek, Romanian, and Slovenian.

Following, the main TV app screenshots are shown:

1. Home screen

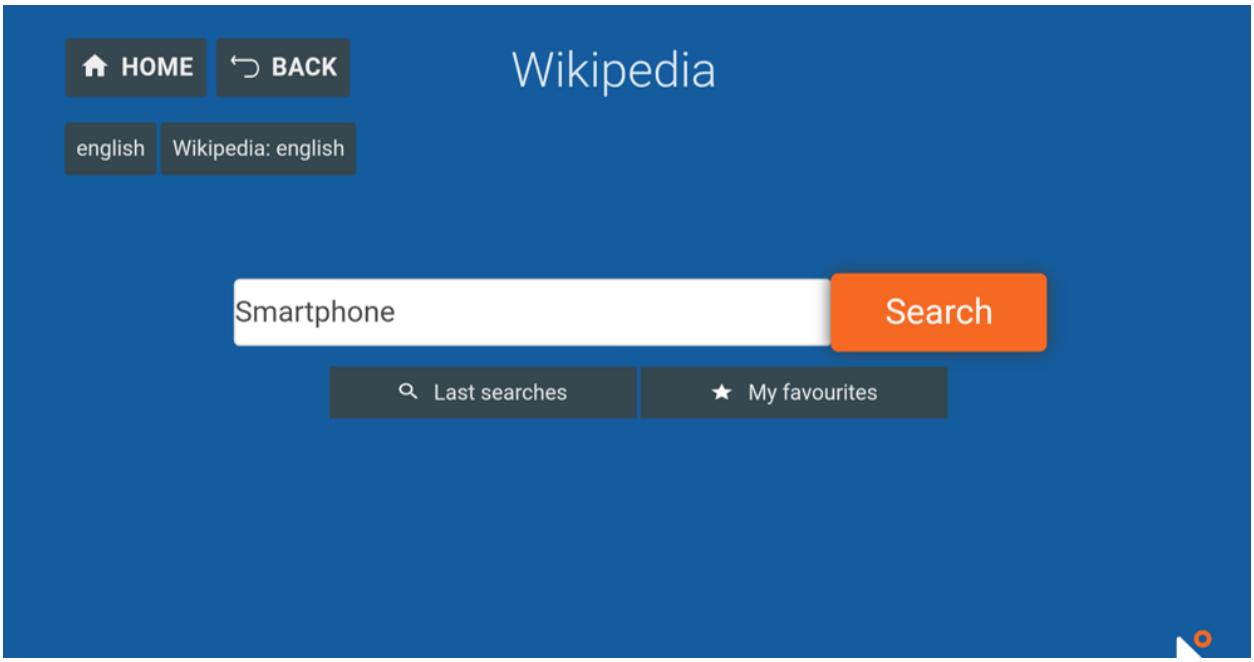


FIGURE 70. WIKIPEDIA HOME SCREEN

2. Search results screen: implements the “Disambiguate result” functionality

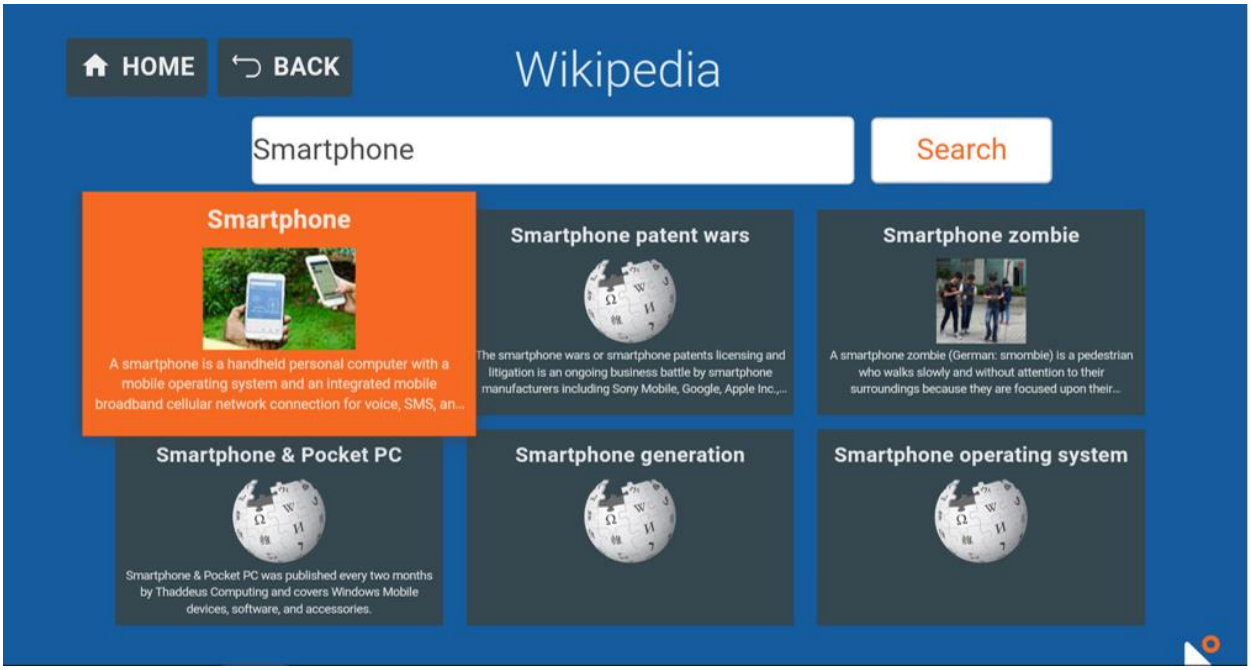


FIGURE 71. WIKIPEDIA SEARCH RESULTS SCREEN

3. Article Screen: shows the selected article content and some buttons to navigate through it.

FIGURE 72. WIKIPEDIA ARTICLE SCREEN

4. Favourites screen

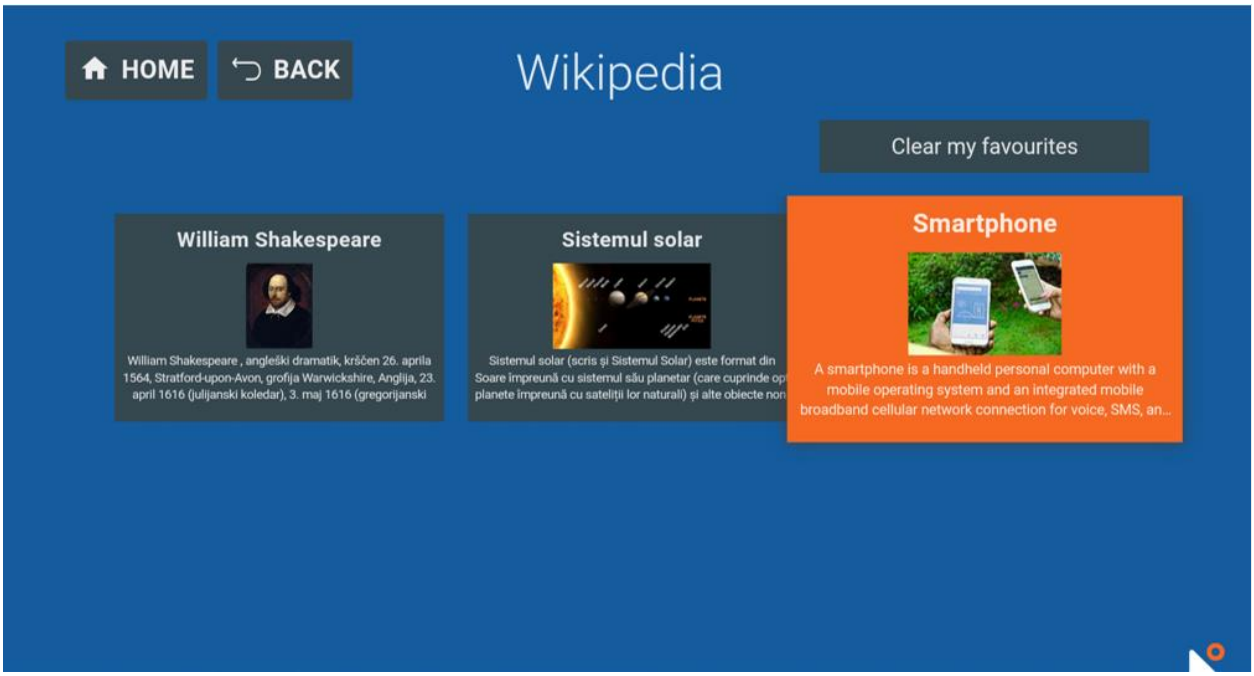


FIGURE 73. WIKIPEDIA FAVORITES SCREEN

5. Last searches screen

FIGURE 74. WIKIPEDIA LAST SEARCHES SCREEN

6. Change language or Change Edition screen

FIGURE 75. WIKIPEDIA CHANGE LANGUAGE OR CHANGE EDITION SCREEN

5.3. SENIOR-TV V3

During the last version of the SENIOR-TV platform the informal services development were focused in **social** and **entertainment** topics: smart leisure, tele-rehab and social nets; in order to improve the relationship of the elderly with their environment and with society.

Furthermore, some services included in previous versions of the platform were upgrade adding new characteristics or fixing detected bugs. In the next sections, the new services and the new characteristics of old services are explained in detail.

5.3.1. Agenda 2.0, Events 3.0, News 3.0, Tracker 2.0 and Wikipedia 2.0

In the SENIOR-TV V3 the Agenda, Events, News, Tracker and Wikipedia services were upgraded to integrate the latest version of the AuthApp updates and to include the Client Ecosystem user roles and relationships (see section 8).

Both, architecture and functionalities, are the same than previous versions but some minor bugs were fixed during this period.

5.3.2. Virtual Center 2.0

In addition of improvements need to integrate the AuthApp new functionalities in the TV app, Virtual Center 2.0 includes a new web app to manage the channels and the user relationships.

The Virtual Center service scope has not changed from previous versions: allow seniors to keep contact with their daily care center from their home through the TV when they should stay at home for any particular reason. The version 2.0 expands the service functionalities to manage the channels in an easy way from a web page. Therefore, the Web app is oriented to the secondary users are responsible to center's broadcasts.

The Virtual Center Web app is accessible from this URL: https://www.imatia.com/seniortv-virtualcenter-web/. The following chart shows the app architecture:

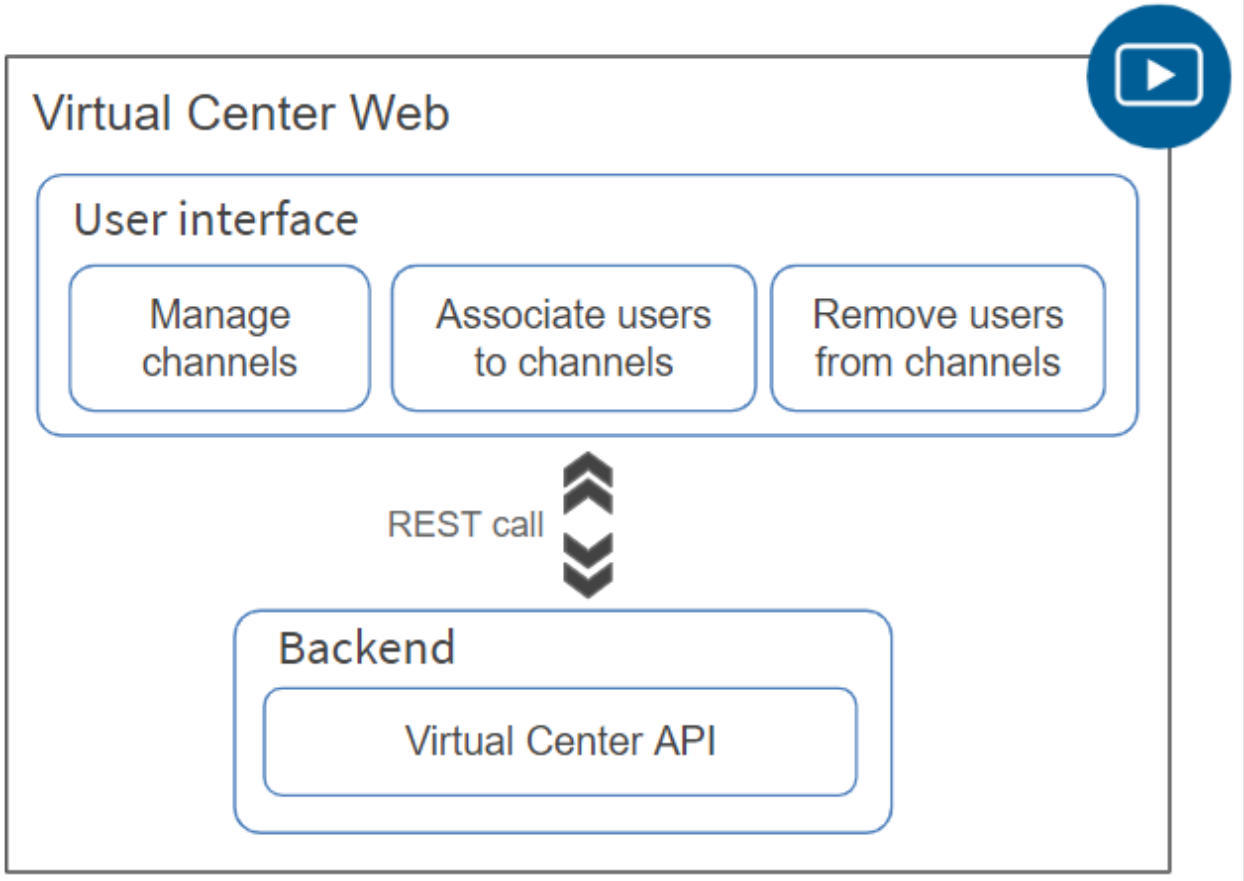


FIGURE 76. VIRTUAL CENTER WEB - ARCHITECTURE

The Virtual center functionalities are the following (below some screenshot are included):

1. Manage channel information: allows secondary users to create, remove and modify the channels' data.
2. Manage users' channels: to organize the seniors' available channels.

The web app was developed using Angular framework and, Java EE technology was used in the cloud backend implementation.

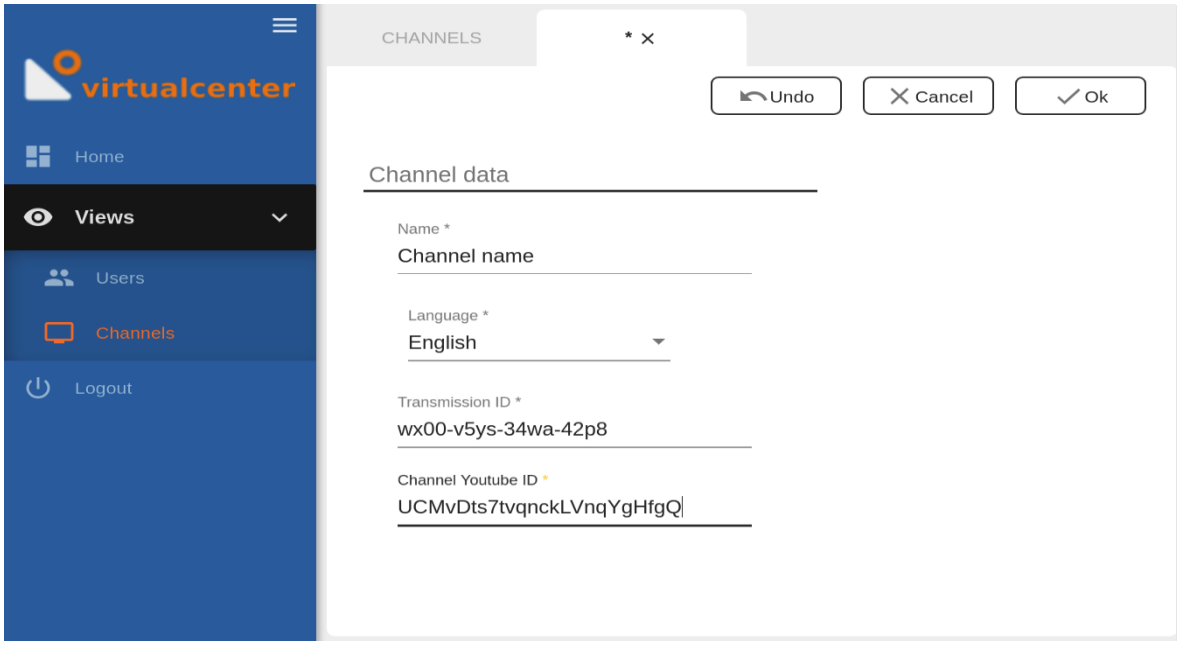


FIGURE 77. VIRTUAL CENTER WEB - NEW CHANNEL

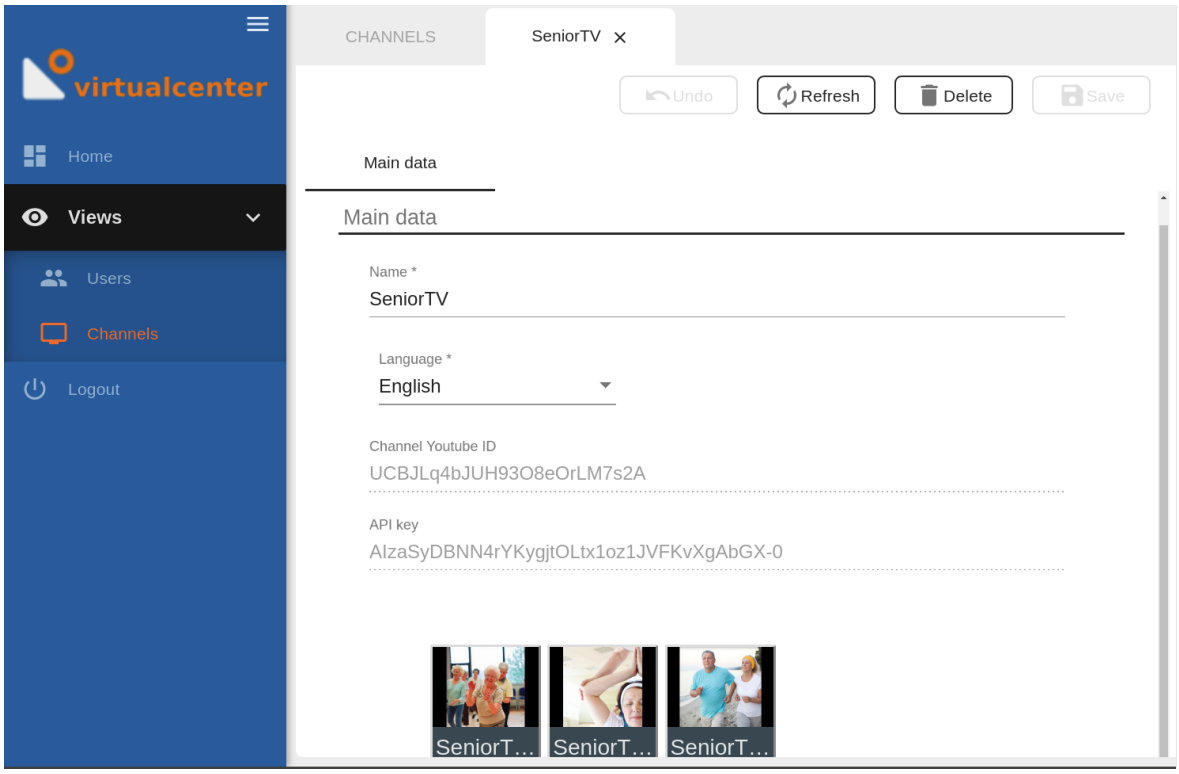


FIGURE 78. VIRTUAL CENTER WEB - EDIT OR REMOVE CHANNEL

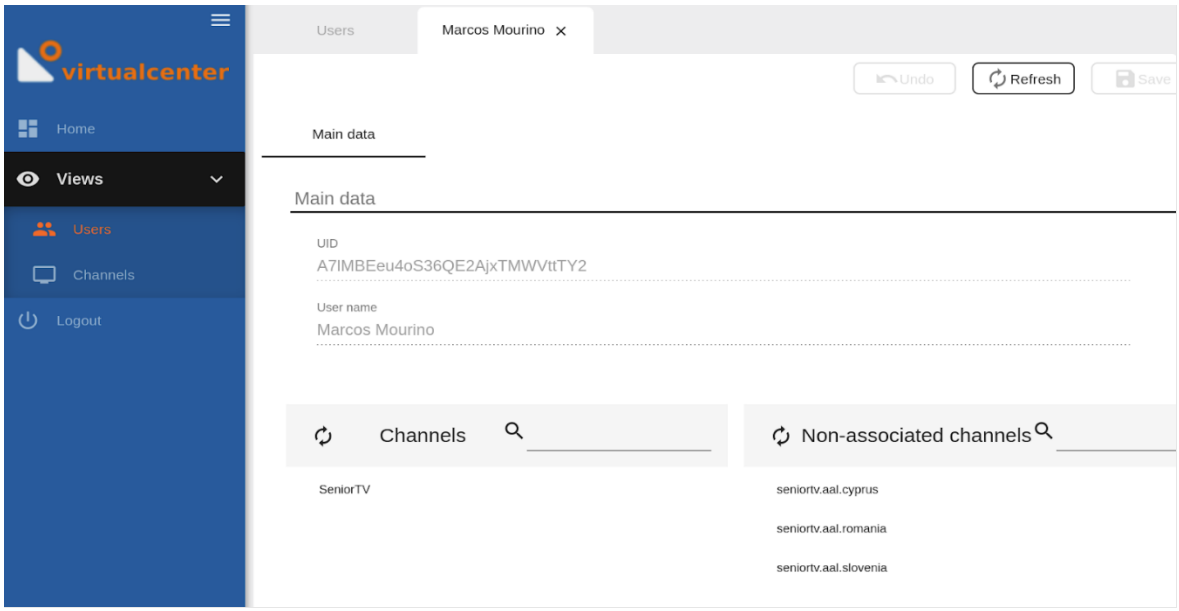


FIGURE 79. VIRTUAL CENTER WEB - ALLOCATE CHANNELS TO USER

5.3.3. Weather 3.0

As previous versions, the Weather 3.0 is composed by two different apps: a TV and a Web app. Some minor developments were included in the TV app to integrate the AuthApp new functionalities. On the other hand, the Web app was upgraded to integrate the user roles and relationships.

5.3.3.1. Weather Web

The Weather Web app allows seniors to manage their favourite locations. This application is focussed to secondary users (caregivers or relatives) but primary users can also access to it. The app has different functionalities in base on user type.

These are the seniors' functionalities:

1. Add a favourite location to a particular user.
2. Remove elements from to a user favourite list.

In addition, the secondary users will have the following functionalities:

3. Check secondary user's seniors
4. Add a favourite location to the selected senior list
5. Remove elements from to a selected senior favourite list

The Weather Web app is accessible from this URL: https://www.imatia.com/seniortv-weather-web/. The next charts show the app architecture and some interface screens:

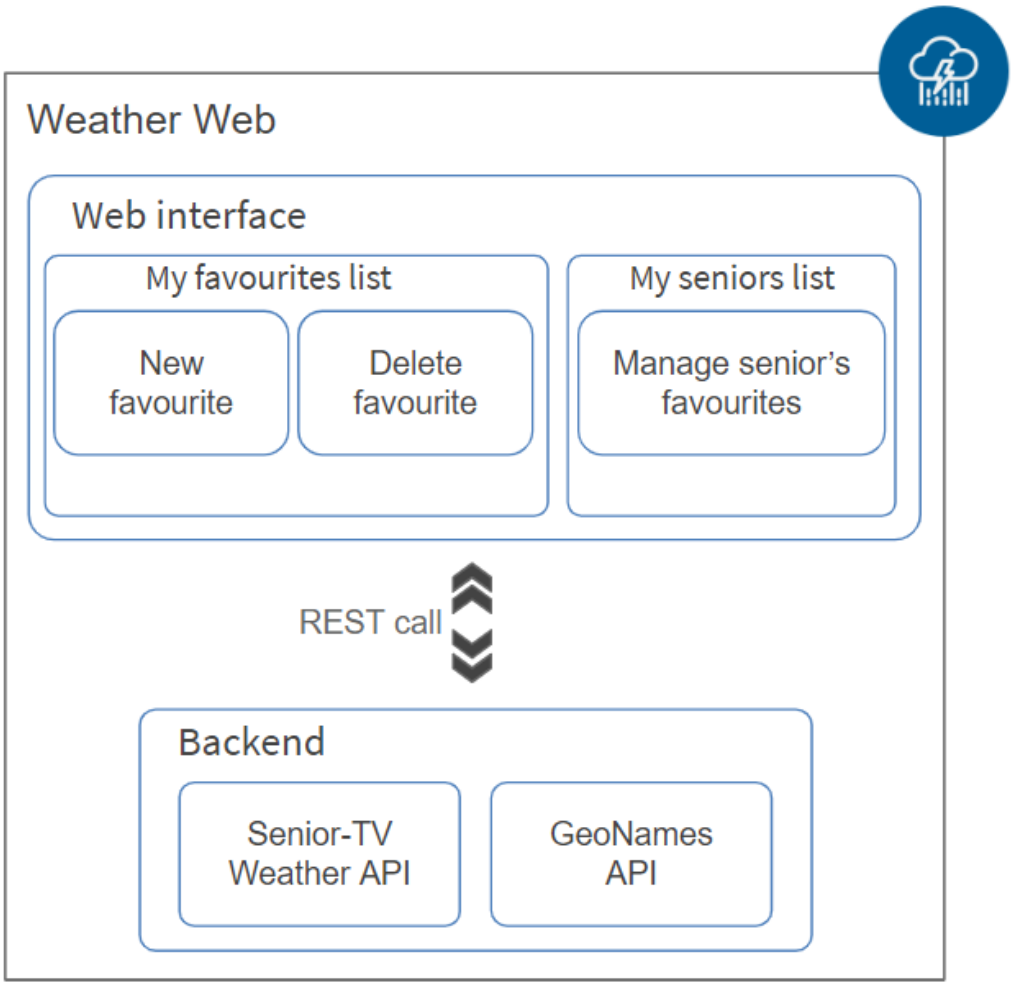


FIGURE 80. WEATHER WEB GENERAL ARCHITECTURE

Secondary and final users can check their own favourite list (see figure 81).

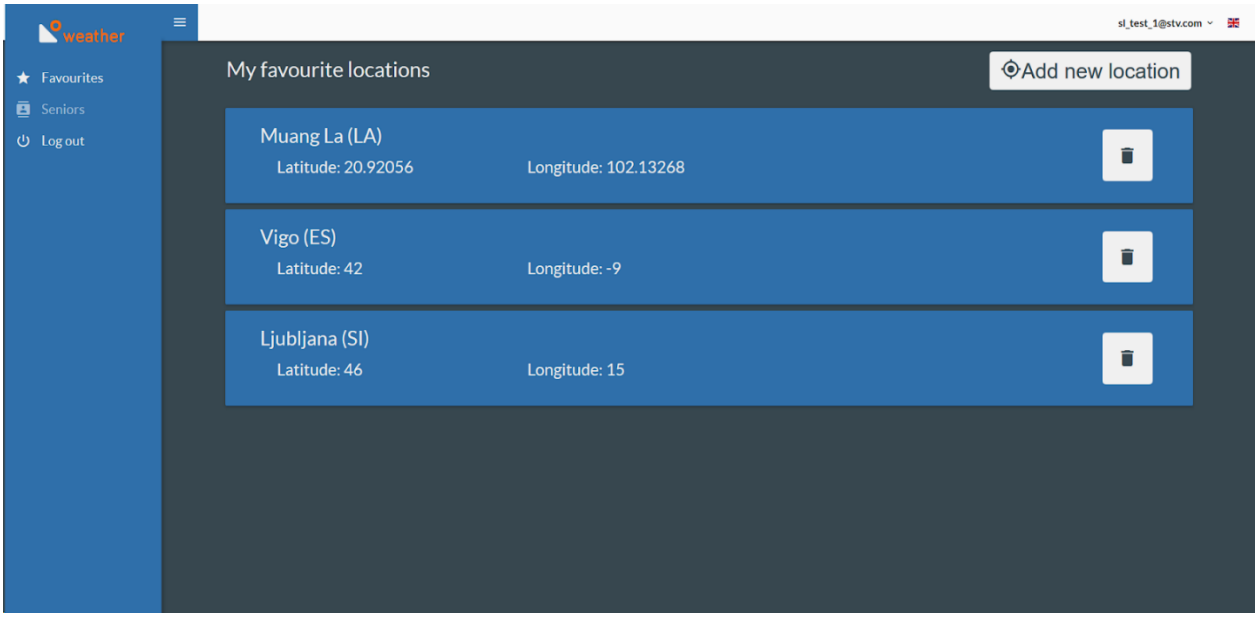


FIGURE 81. WEATHER WEB - SECONDARY USER INTERFACE

In addition of the favourite lists, secondary users can manage their seniors' favourite lists. This functionality is accessible from the "Seniors" button of left menu. Figure 82 shows the seniors related to a secondary user. When the user presses the "Edit" button of a senior card, the senior's favourites are shown.

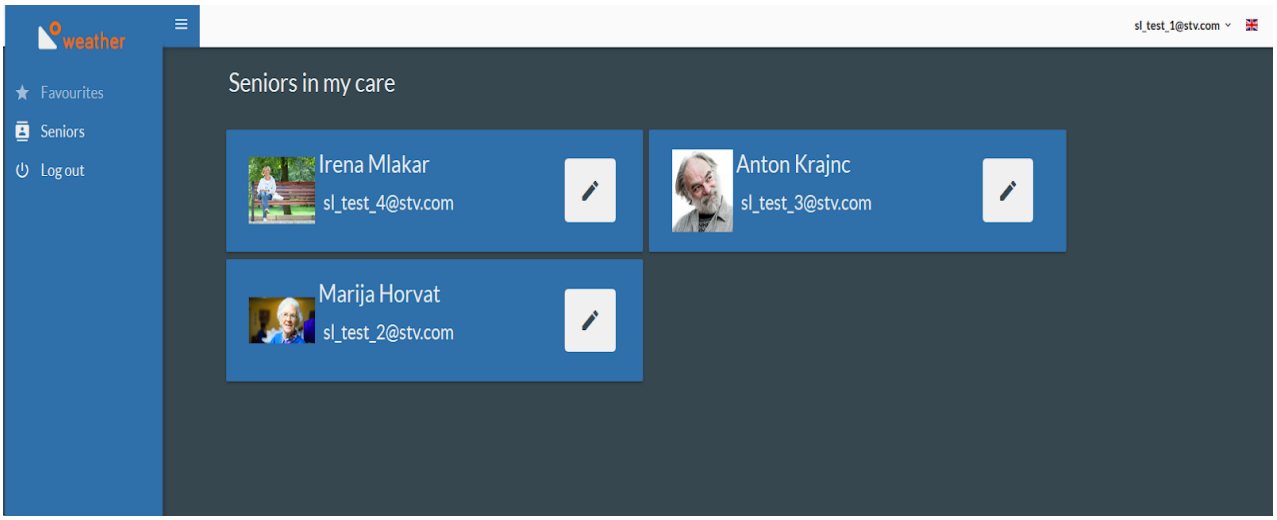


FIGURE 82. WEATHER WEB - SENIOR LIST

5.3.4. Audiovisual Channels 1.0

The Audiovisual Channels service allows seniors to access to a categorized and multi-thematic multimedia repository using their TVs at home. The repository can contain videos on several topics called channels: physical exercises, rehabilitation exercises, cooking recipes, etc. These channels serve as informal learning

elements for seniors. The videos can belong to more than one channel and users can be registered in several channels, offering a personalized experience.

The Audiovisual Channels service is composed by two different apps: a TV and a Web app. The TV app is focused on seniors, and the Web app is oriented to secondary users. The following chart shows the architecture of the service:

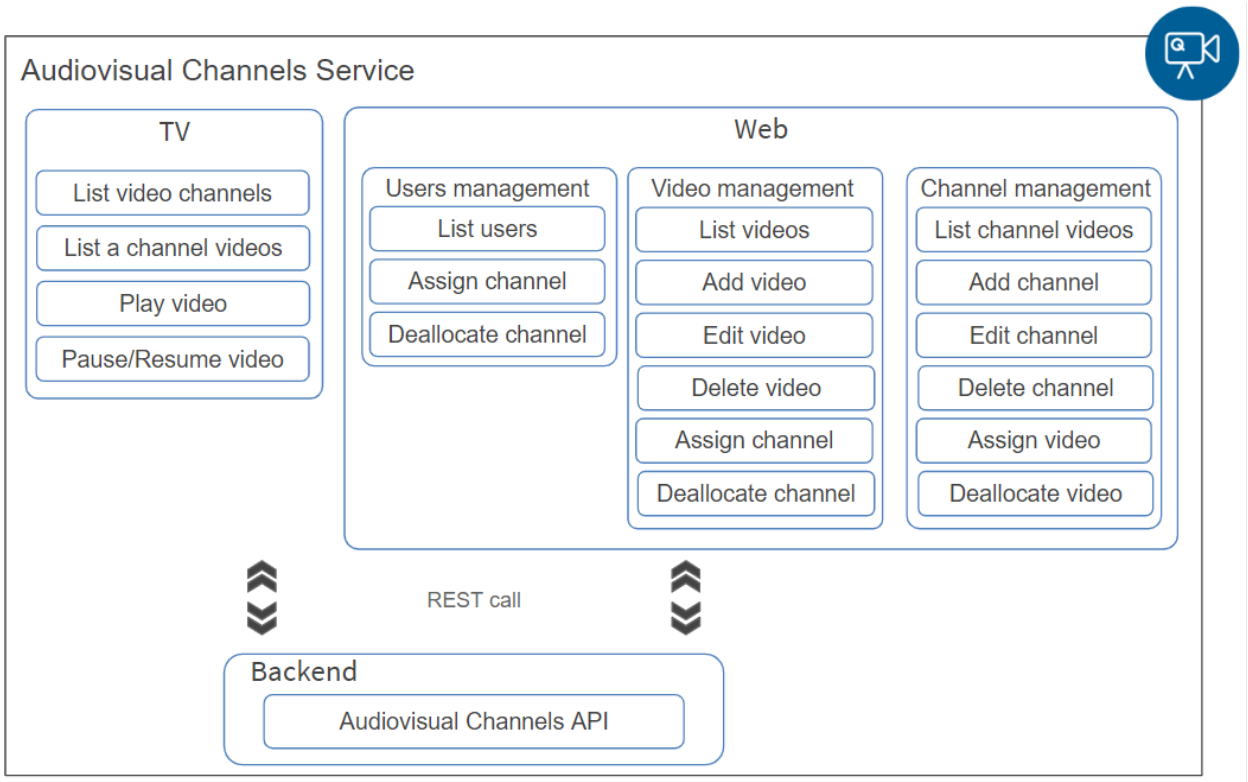


FIGURE 83. AUDIOVISUAL CHANNELS - ARCHITECTURE

Both, Web and TV apps, have been implemented using Angular framework and Apache Cordova for TV app. The two apps use HTTP request to communicate with the backend API to obtain the channels information. The cloud backend has been developed using JavaEE/Spring technology. Audiovisual Channels API allows manage the videos, channels, and the relationships between them and the users. The next paragraphs include the TV and Web applications architecture and the main interface screens.

5.3.4.1 Audiovisual Channels TV

The TV application is oriented to seniors in order to allow them to access to the multimedia content of the repository in a simple way. The main functionalities are listed below.

1. List channels: allows users to see the list of available video channels associated to the user.
2. List a channel videos: allows users to see the list of videos included in the selected channel.
3. Play videos: allows users to play a specific video.

4. Pause/Resume video: allows users to pause and resume playback of a video that is playing.

The following charts show the Audiovisual Channels TV app architecture and some interface screens:

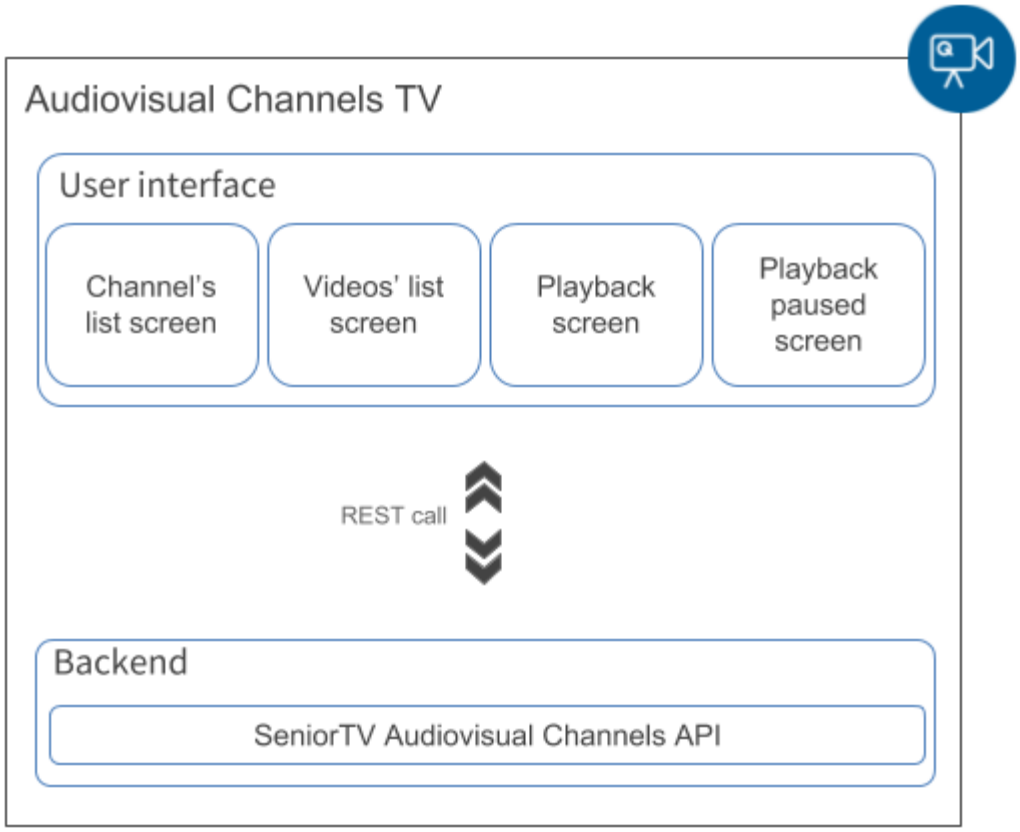


FIGURE 84. AUDIOVISUAL CHANNELS - TV APP ARCHITECTURE

1. List Channels screen

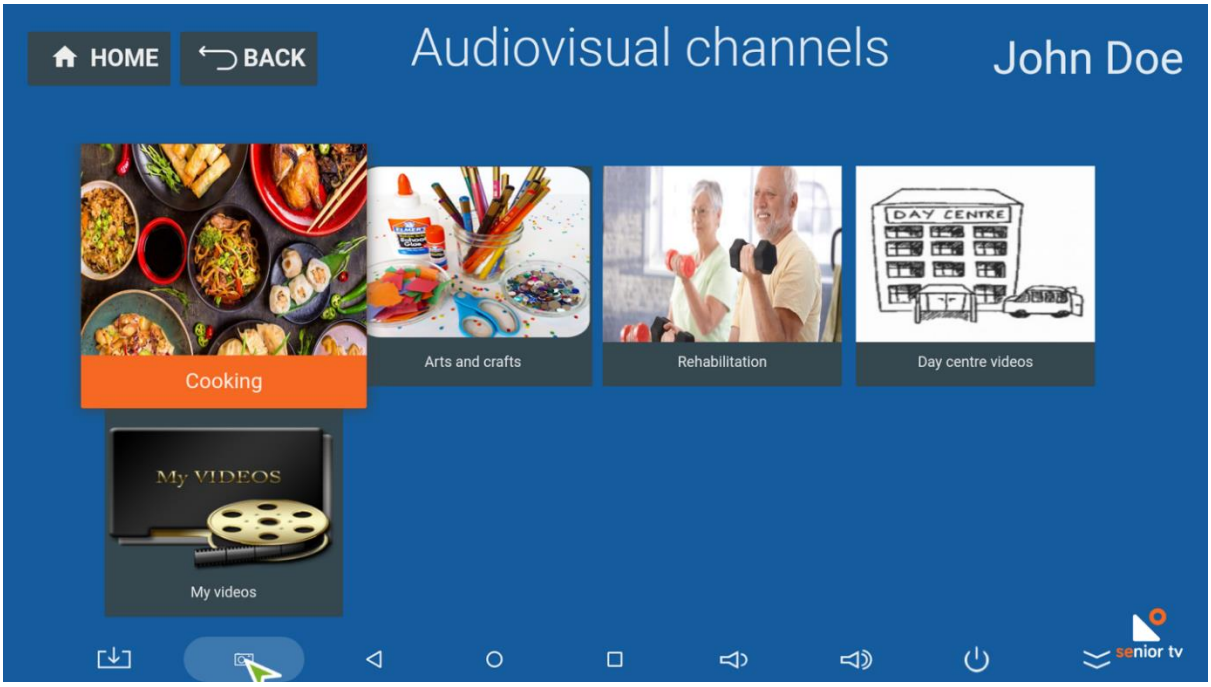


FIGURE 85. AUDIOVISUAL CHANNELS - CHANNELS

2. List Videos screen

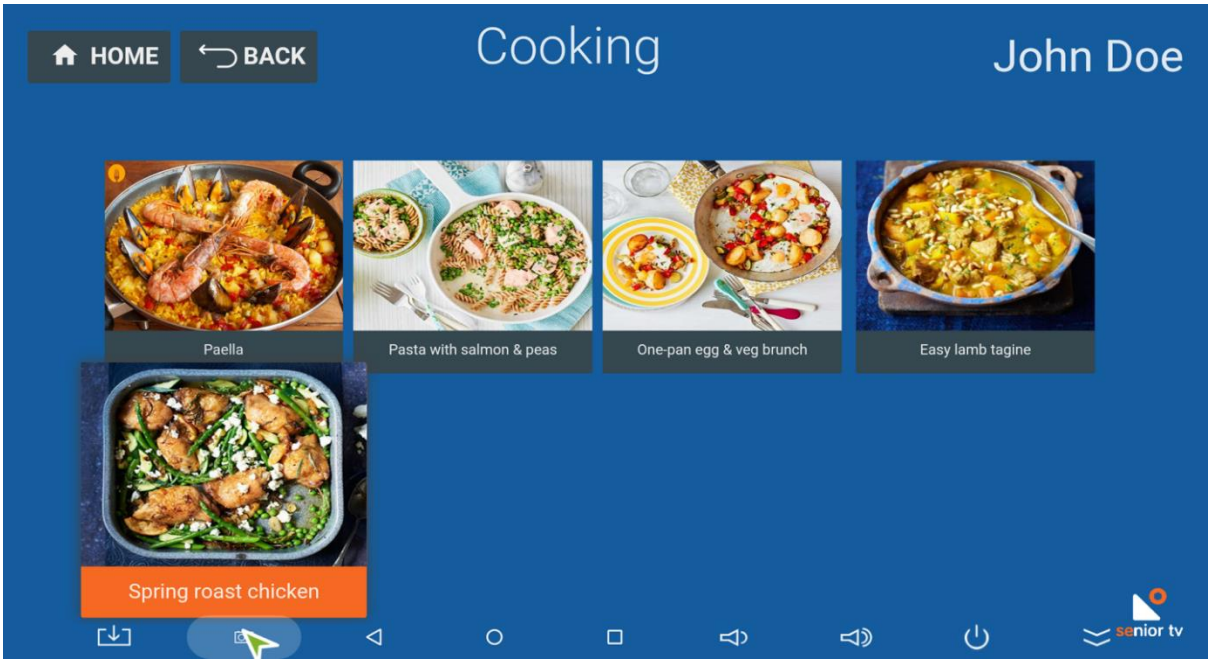


FIGURE 86. AUDIOVISUAL CHANNELS – VIDEOS

3. Playback screen

FIGURE 87. AUDIOVISUAL CHANNELS – PLAYBACK PAUSE

5.3.4.2. Audiovisual Channels Web

The Web application is oriented to secondary users, so that, they can manage the videos, the channels and the relationships between videos, channels and users. The main functionalities are listed below.

1. Manage channels: allows secondary users to create new channels and list, remove and edit existed channels.
2. Manage videos: allows secondary users to upload new videos to the repository, and list and remove existed ones. In addition, videos can be associated to one or more channels.
3. Manage users: allows secondary users to organize the available channels for each senior considering their preferences.

The Audiovisual Channels Web app is accessible from this URL: https://www.imatia.com/seniortv-channels-web/. The next charts show the app architecture and some interface screens:

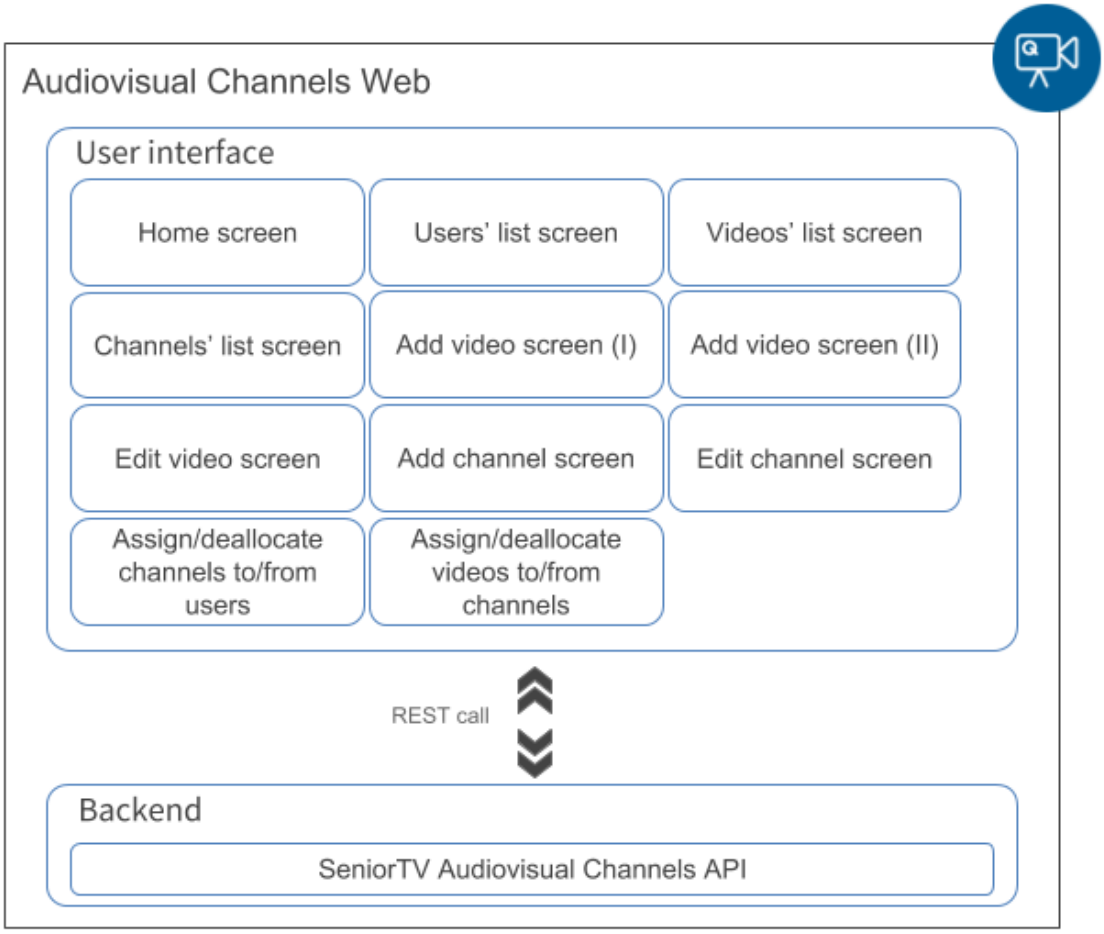


FIGURE 88. AUDIOVISUAL CHANNELS - WEB APP ARCHITECTURE

1. List channels screen

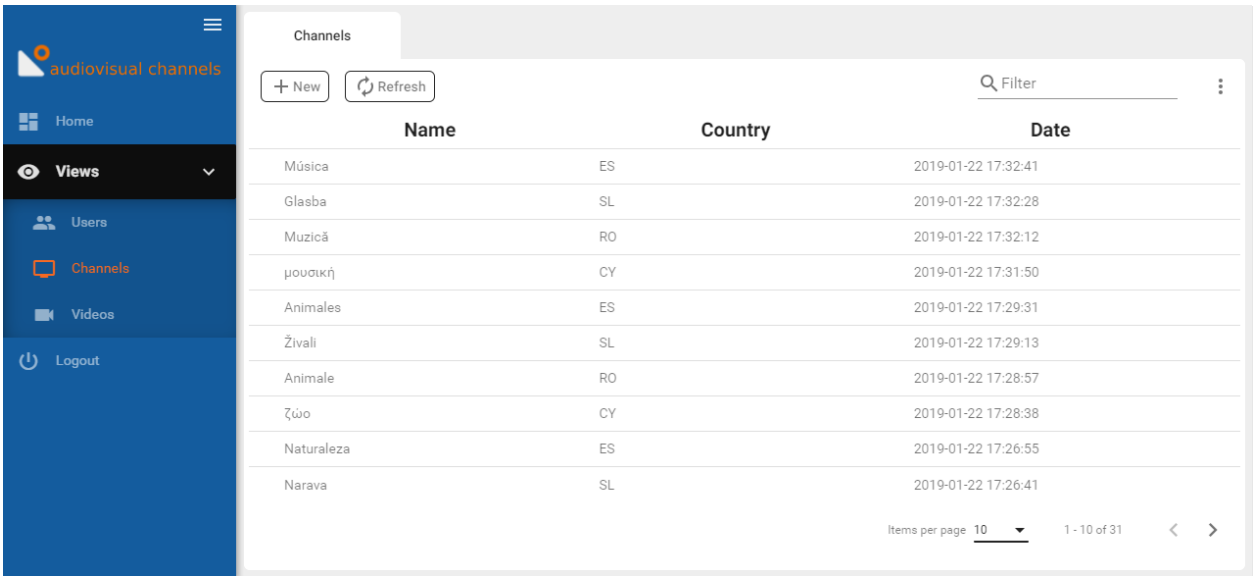


FIGURE 89. AUDIOVISUAL CHANNELS - WEB LIST CHANNELS

2. Edit channel & associate videos screen

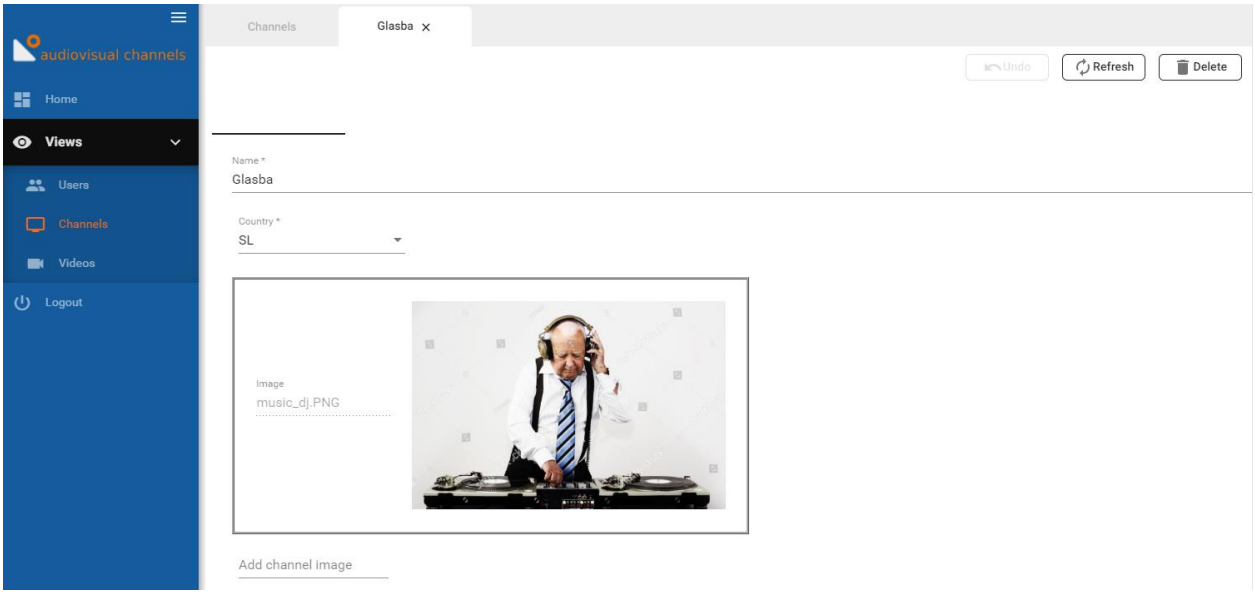


FIGURE 90. AUDIOVISUAL CHANNELS - WEB NEW CHANNEL

3. List videos screen

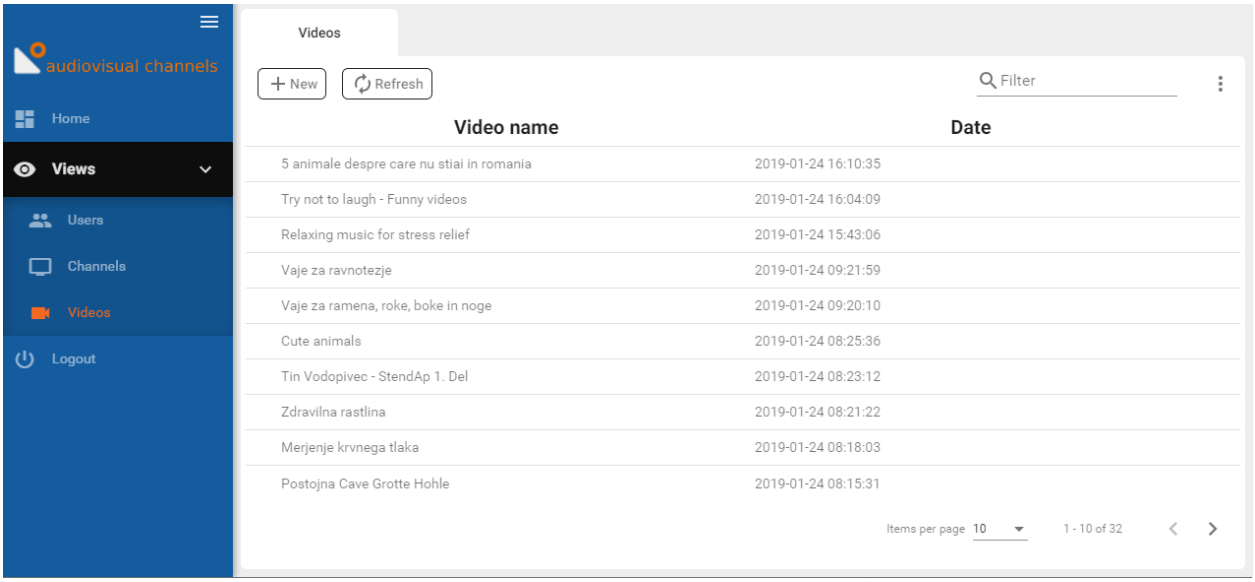


FIGURE 91. AUDIOVISUAL CHANNELS - WEB LIST VIDEOS

4. Upload a video screen

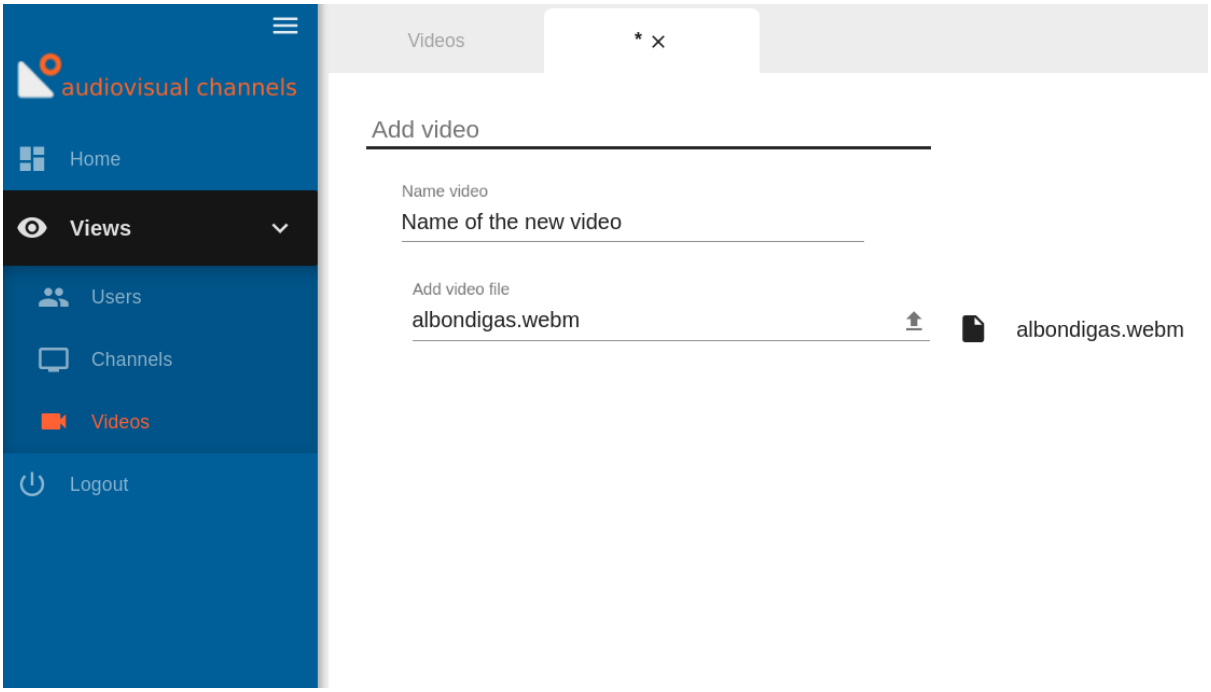


FIGURE 92. AUDIOVISUAL CHANNELS - WEB APP UPLOAD A VIDEO

5. Edit a video screen

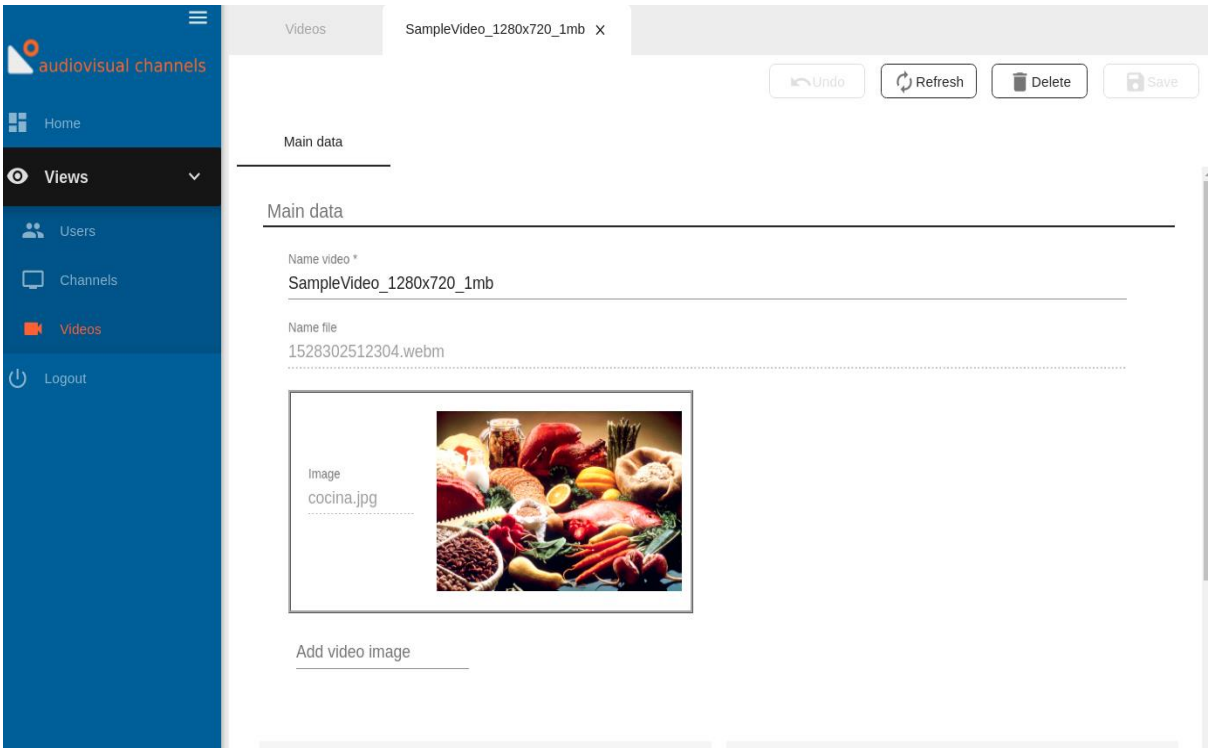


FIGURE 93. AUDIOVISUAL CHANNELS - WEB EDIT VIDEOS

5.3.5. Video chat 1.0

This service is designed to keep older people in touch with their friends, family and caregivers. It allows older people to close gaps in distance and technology and communicate with their contacts quickly and easily. To start receiving video calls, seniors only need to connect a webcam and a microphone to their devices.

The service allows seniors to start and receive video calls connecting a camera and a microphone to their devices. The Video Chat is divided in two main modules. The chart in figure 94 shows the architecture and modules of the service:

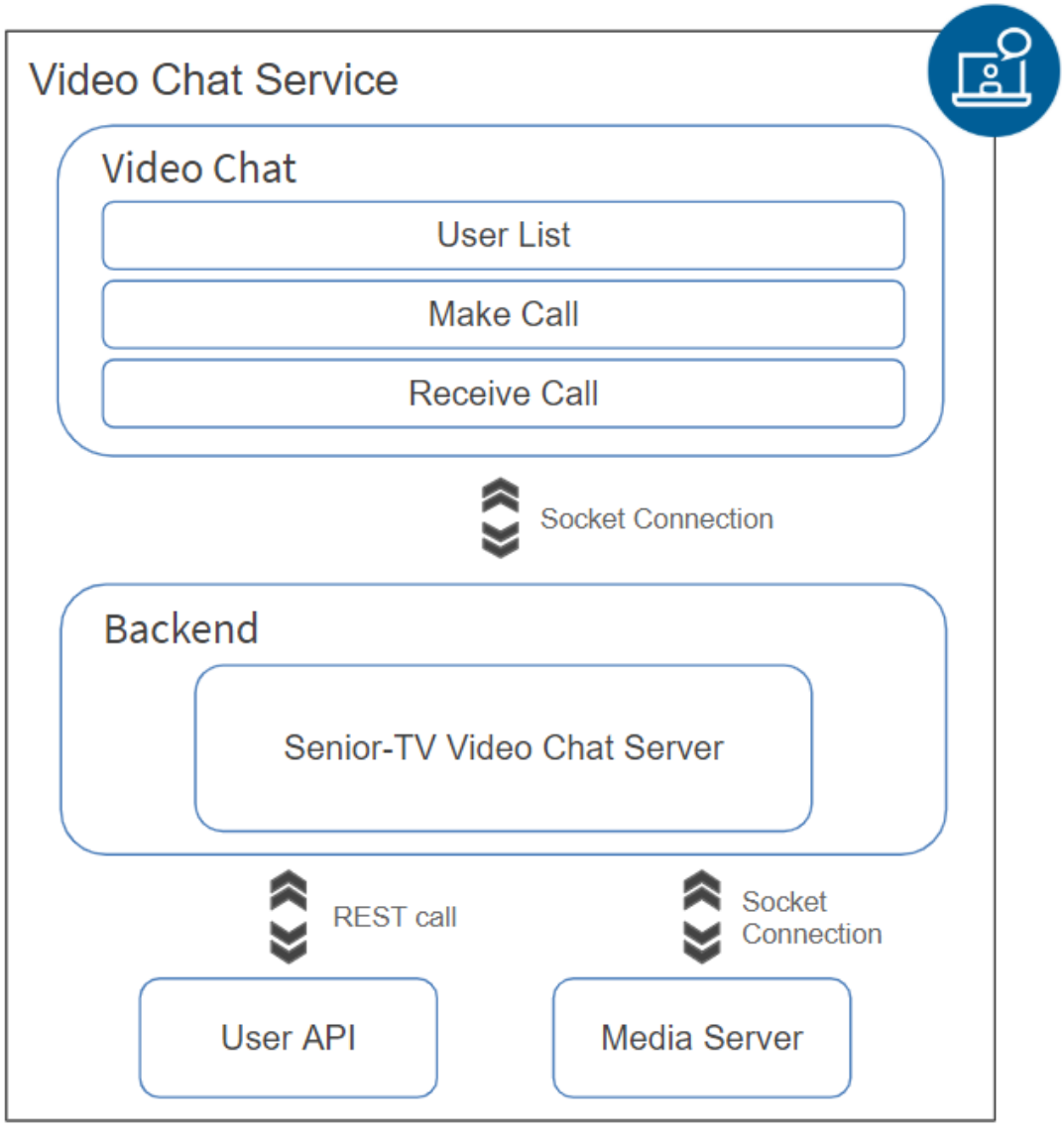


FIGURE 94. WEATHER WEB - SENIOR LIST

The TV app has been developed with Angular and it has been packaged with Apache Cordova to create the Android app. On the other hand, Video Chat Backend has been developed with Java EE/Spring, using a Spring architecture and it uses the *Kurento Media Server* technology to perform audio and video calls.

Kurento Media Server is an open source software based on WebRTC architecture. WebRTC is an open project that provides Real-Time Communications (RTC) capabilities via simple APIs to browsers and mobile applications.

As the previous diagram shows, the Video Chat backend is connected with the TV app using websockets technology. Websocket is a communication protocol which provides full-duplex communication channels. The channels allow to keep opened connections between two sides and passing messages between them. The passed messages include information about the users online/offline status, incoming and outcoming calls and WebRTC negotiation parameters.

The websocket technology is also used to communicate Video Chat backend and Kurento service to create and destroy chat channels and, to send the audio and video data between users.

The following paragraphs explain the Video Chat architecture in detail. The server side consists in three different sections: Video Chat server, Media server and User API.

- Video Chat server is the main element of the backend and it has two key objectives:
    1. Provide the information of friend list to the connected users. The user information includes the name, the photo and the status connection. Online users have to know in real time when their friends become online or offline to be able to call them.
    2. Manage the initial and final steps of video chat conversation. These steps include:
        a. Alert user to answer or hang up an incoming call.
        b. Manage call status to auto-hang up when user is unavailable and he's not answering while the other user is waiting for a response. This server has the objective of negotiate the parameters of the call with the media server and the Android devices, and then send all the necessary information to the Video Chat app to start or cancel the call.
        c. Finish calls when the connection is lost or user hangs up the current call.
- Media server is the responsible of hosted the video calls, it is the responsible of managing pipes to establish the calls. Once the pipes are configured, the Media server interchange the media information between the two call sides.

  Video Chat TV app, Kurento and the Video Chat server use messages to communicate one to each other. These messages are sent through opened websockets.

- Users API: is the responsible of manage the data of SENIOR-TV platform users (see section 8.1.4.). The Video Chat Server will request the friends' information to Users API.

The Video Chat TV app will be used by the seniors to interact with the Video chat service through their TVs. Users will have the classic telephone functionalities watching their friends live on TV:

1. Check the friend's status (online/offline).
2. Manage calls
   - Call an online friend (figure 95).
   - Accept or decline incoming calls (figure 96)
   - Hang up a call (figure 97).

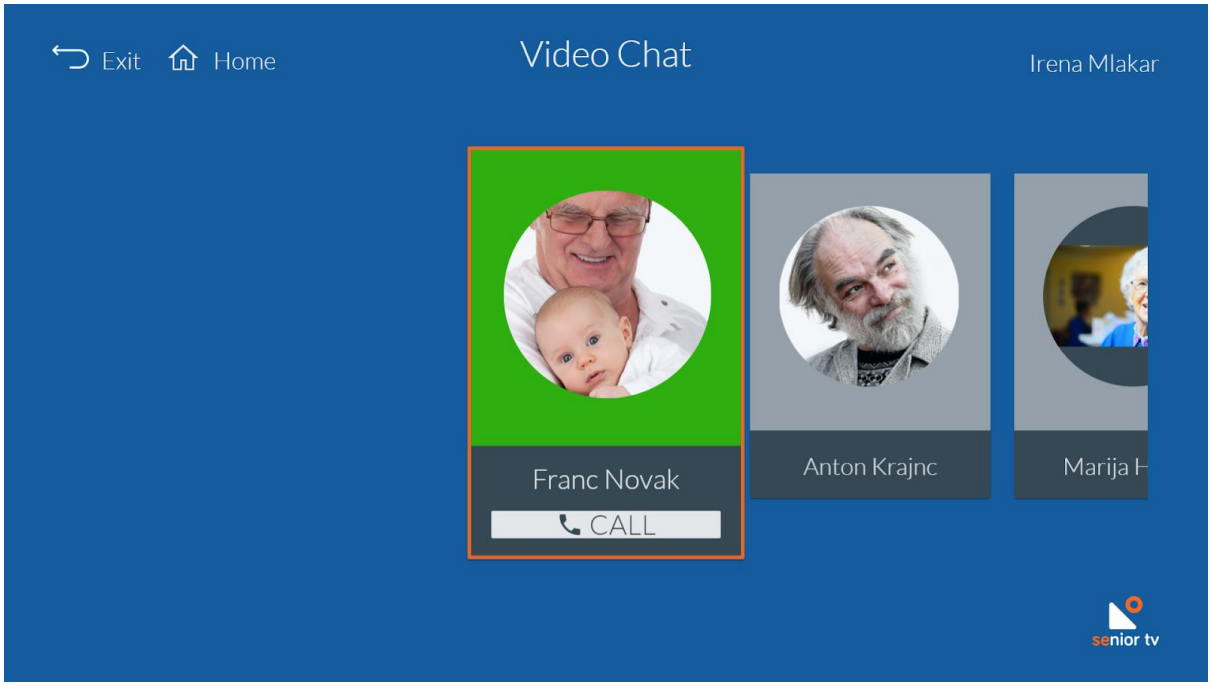


FIGURE 95. VIDEO CHAT – FRIENDS LIST

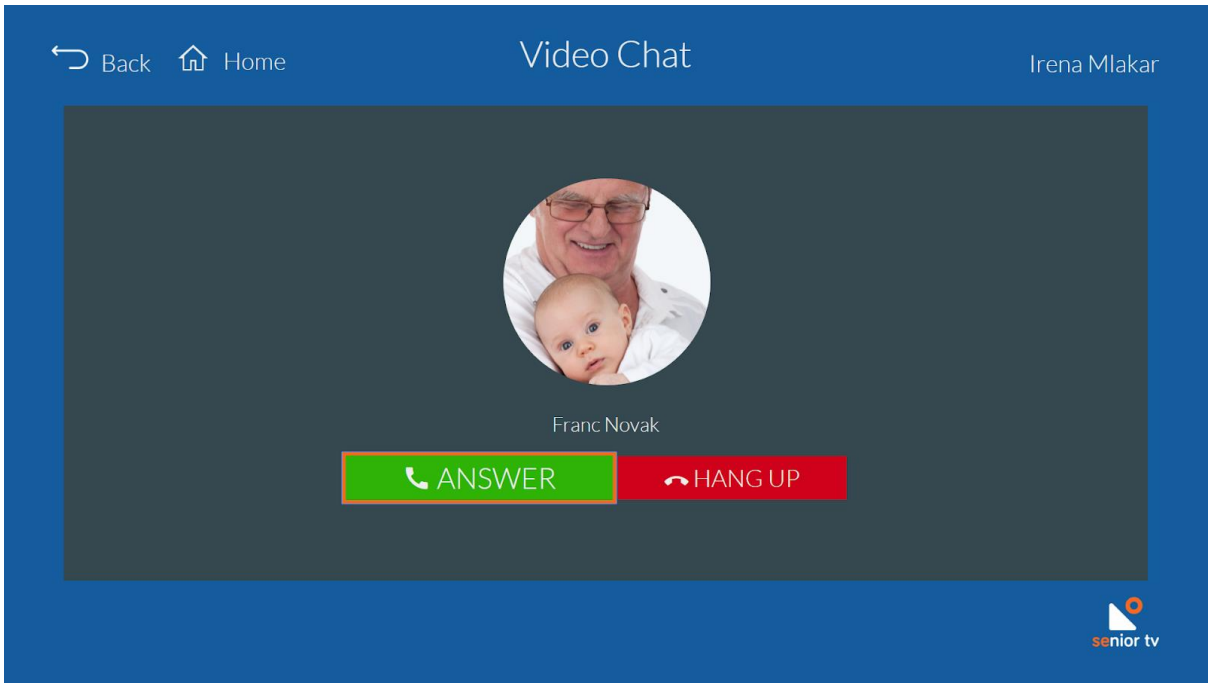


FIGURE 96. VIDEO CHAT – RECEIVING CALL

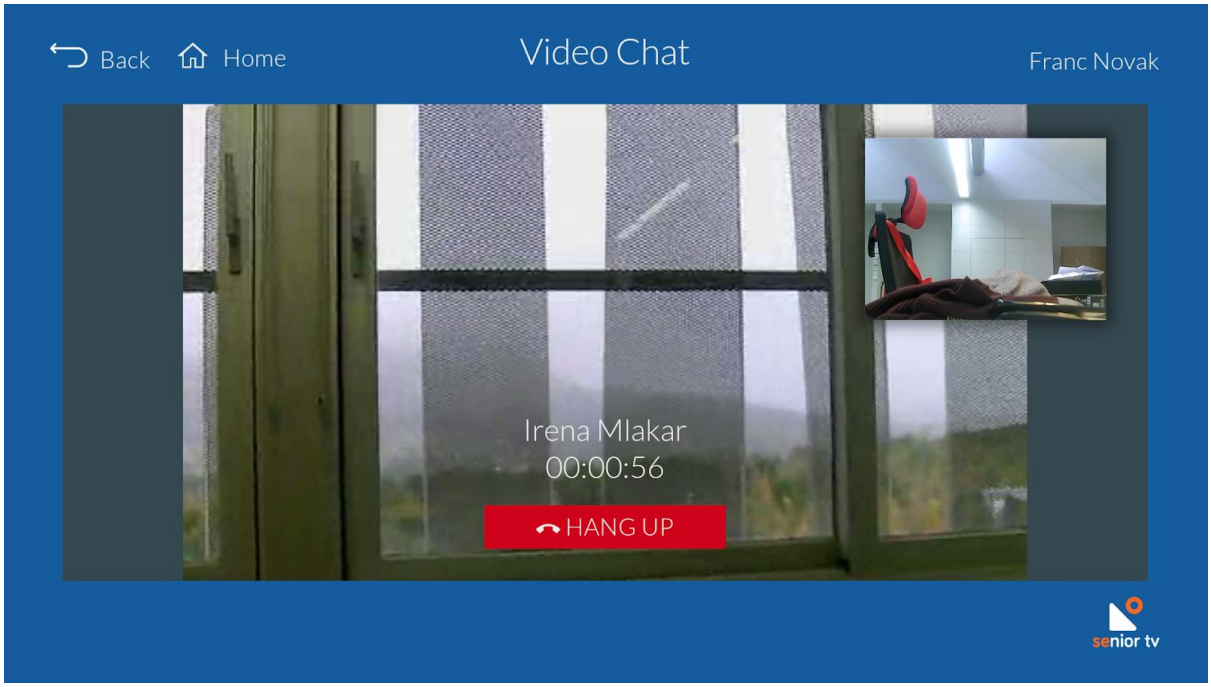


FIGURE 97. VIDEO CHAT – CALL

5.3.6 Social Nets 1.0

The Social Nets service has been designed to help older people to interact with new technologies, in particular, the today most used social networks: Facebook, Instagram and Twitter.

The aim of the service is presenting the social networks capacities in an easy way, reducing the multiple options available in a conventional social network application to two type of functionalities: view my profile data (followers, friends and general information) and see my publications.

Social Nets service is composed only by an TV app which uses the social network APIs to access to the user's data. The service architecture is detailed in the following chart:

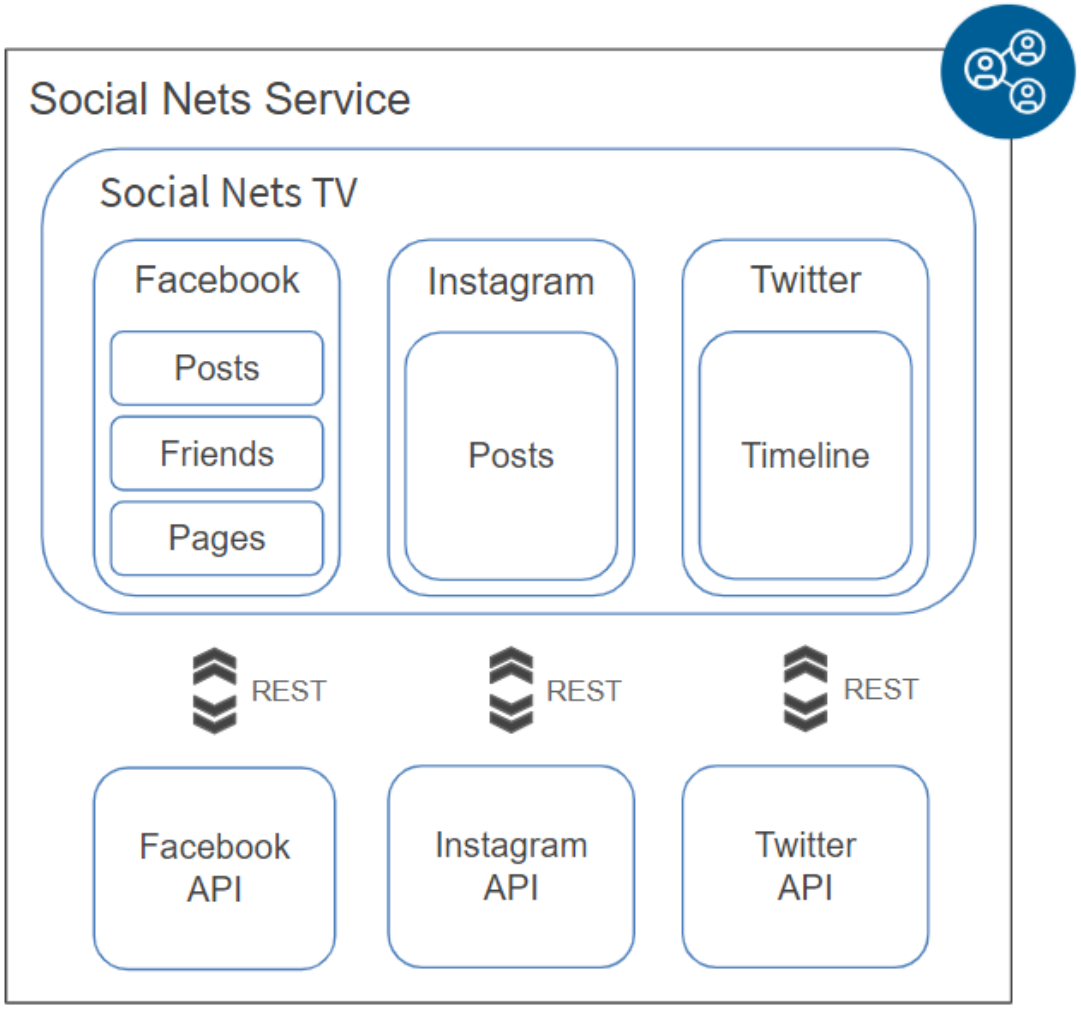


FIGURE 98. SOCIAL NETS ARCHITECTURE

The Social Nets app has been developed with in Angular and packaged with Apache Cordova. The app uses REST technology to access to the three social networks data. In the case of Twitter and Facebook, two Cordova plugins have been used to manage the requests. It is not the case of Instagram; which requests are made from the Angular code directly.

In order to access to a social network information, it is mandatory start a session with a social user credentials. After logging in, the user session will be identified with a "token". This token is essential to

make any request to the social network APIs. The token identifies the user who makes the requests and the information available for him/her.

Despite of the three nets are integrated in the same SENIOR-TV service, they are independent, which means the token used by one net is not valid to another one. Therefore, it was necessary included three different login processes, one for each of the nets integrated in the service. On the other hand, users do not need to have a user for all the nets and they will be able to access to the nets that they want. The following figure shows the main screen of the Social Nets app:

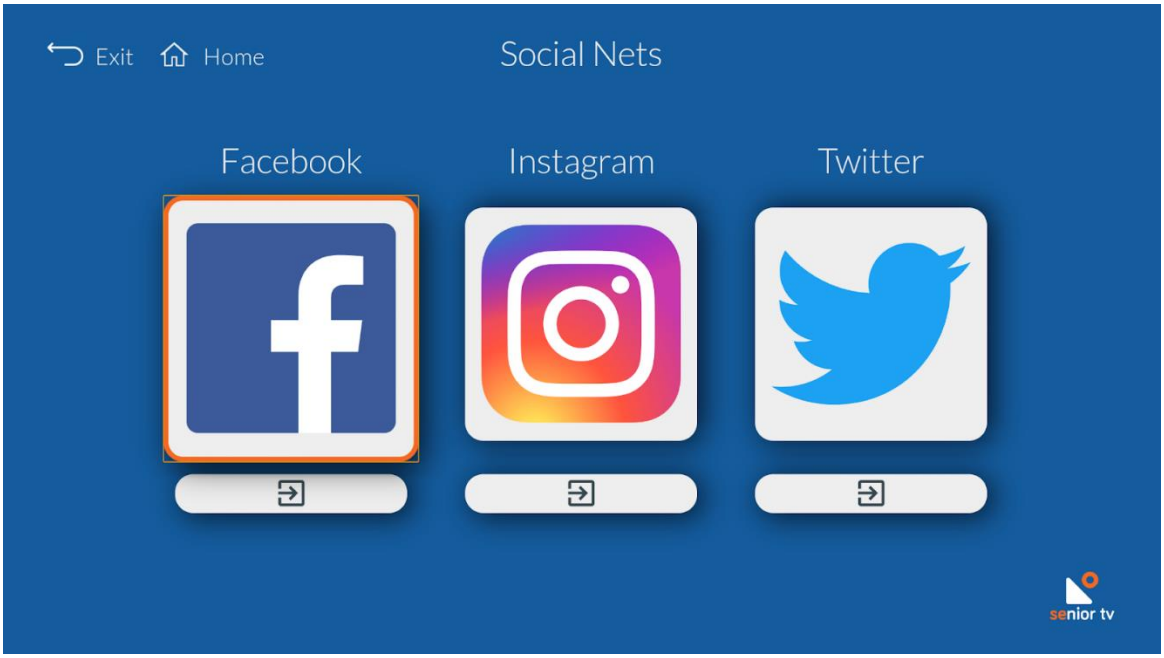


FIGURE 99. SOCIAL NETS - MAIN SCREEN

From the *Main screen*, seniors will be able to login and logout to each of the social network using their own credentials. After first login, it will not necessary indicate the credentials again. The information shown in the *Posts screen* will be slightly different depending on the social network:

- Facebook: this is the most complex API due to the multiple possibilities that it offers. Social Nets service includes only three main functionalities in order to present the information in an easy way and simplify the interface navigation:
  1. Check the user latest posts (figure 100).
  2. Check the list of friends.
  3. Check the followed Facebook pages (figure 101).
- Instagram: the requests made to this API were reduced to the following (figure 102):
  1. Get user data: name, total posts, number of followers and following users.
  2. Get user posts: multimedia information (pictures and videos), comments and likes.

- Twitter: to access to this API information a Twitter Cordova plugin was used. The plugin allows Angular apps to interact with Twitter Android SDK which made the API requests. The obtained data includes the following (figure 103):
  1. User data: name, total tweets, number of followers and following users.
  2. User tweets: multimedia information (pictures and videos), text, number of favourites and retweets.

  All this information is not accessible from the Cordova plugin directly and it was necessary to modify the plugin to obtain some of these fields.

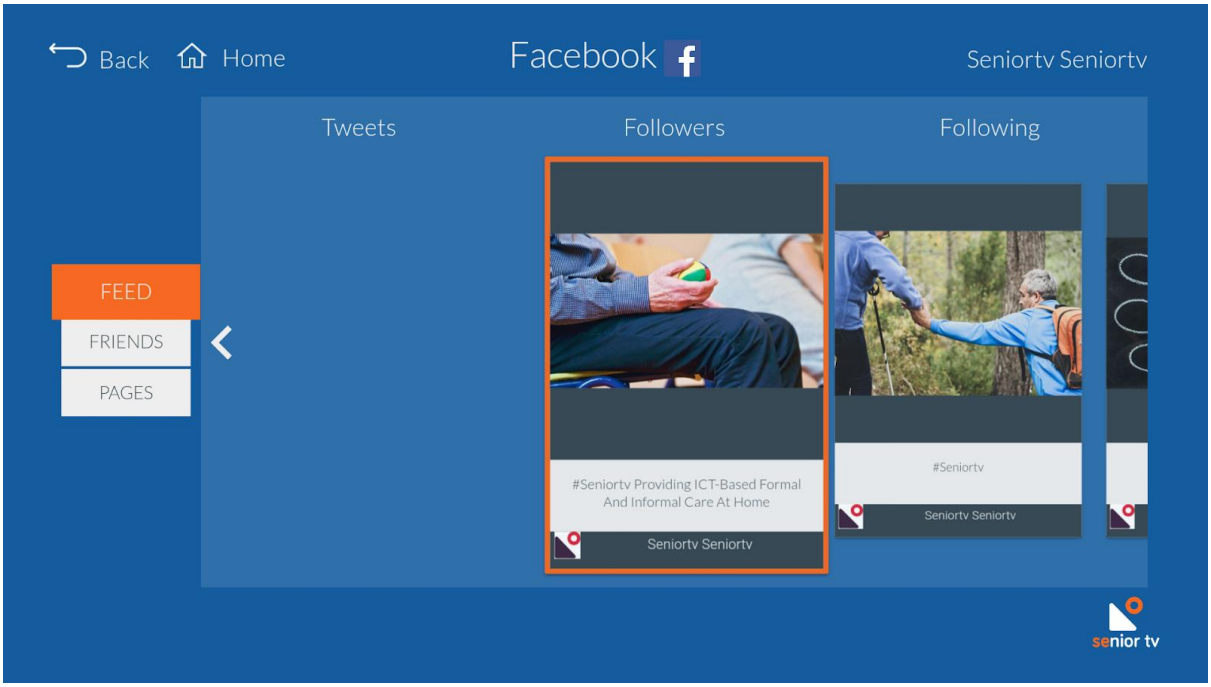


FIGURE 100. SOCIAL NETS - FACEBOOK POSTS

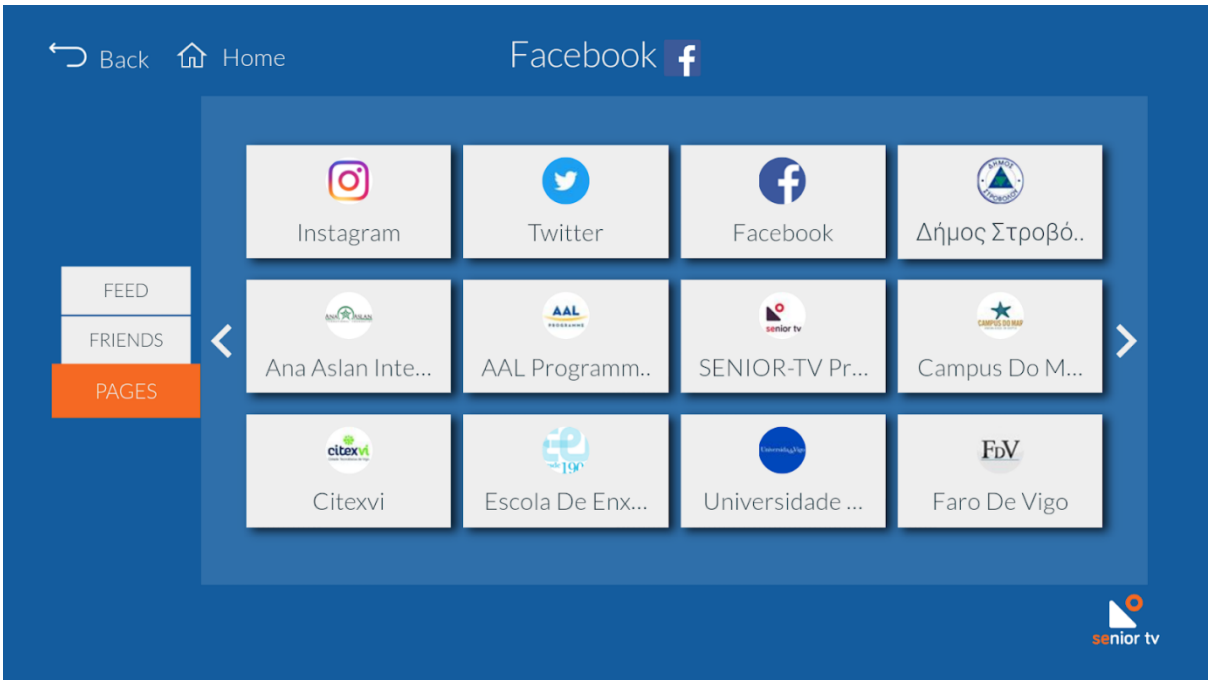


FIGURE 101. SOCIAL NETS - FACEBOOK FOLLOWED PAGES

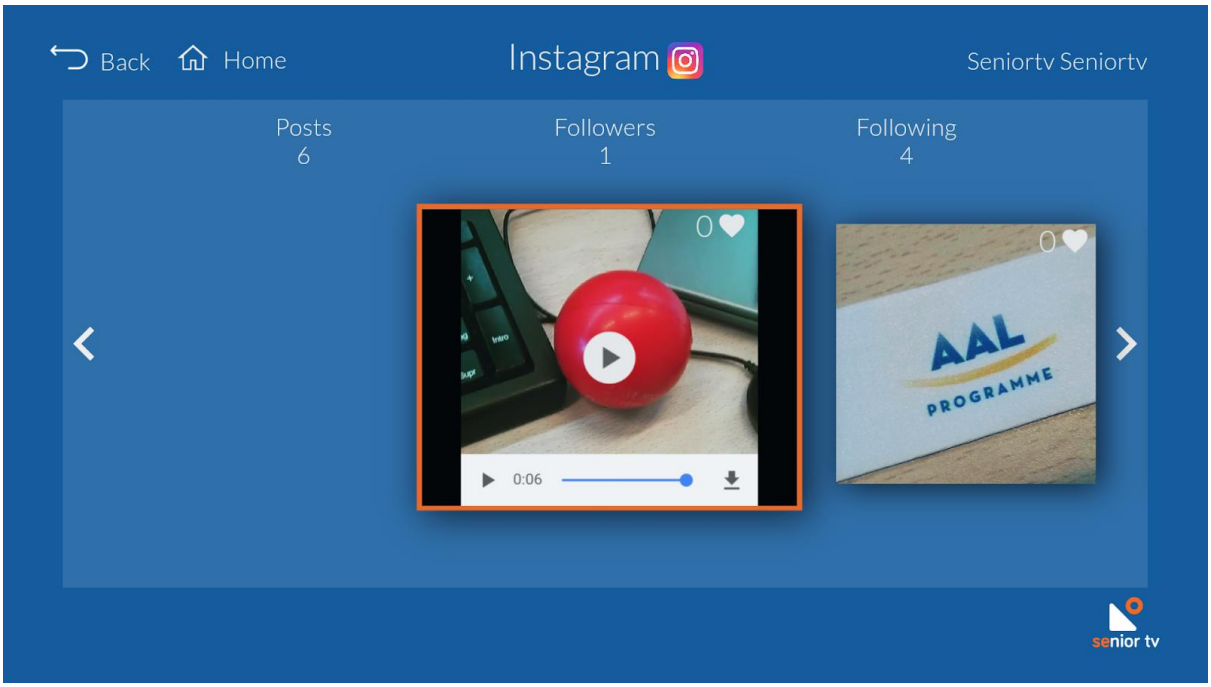


FIGURE 102. SOCIAL NETS - INSTAGRAM POSTS

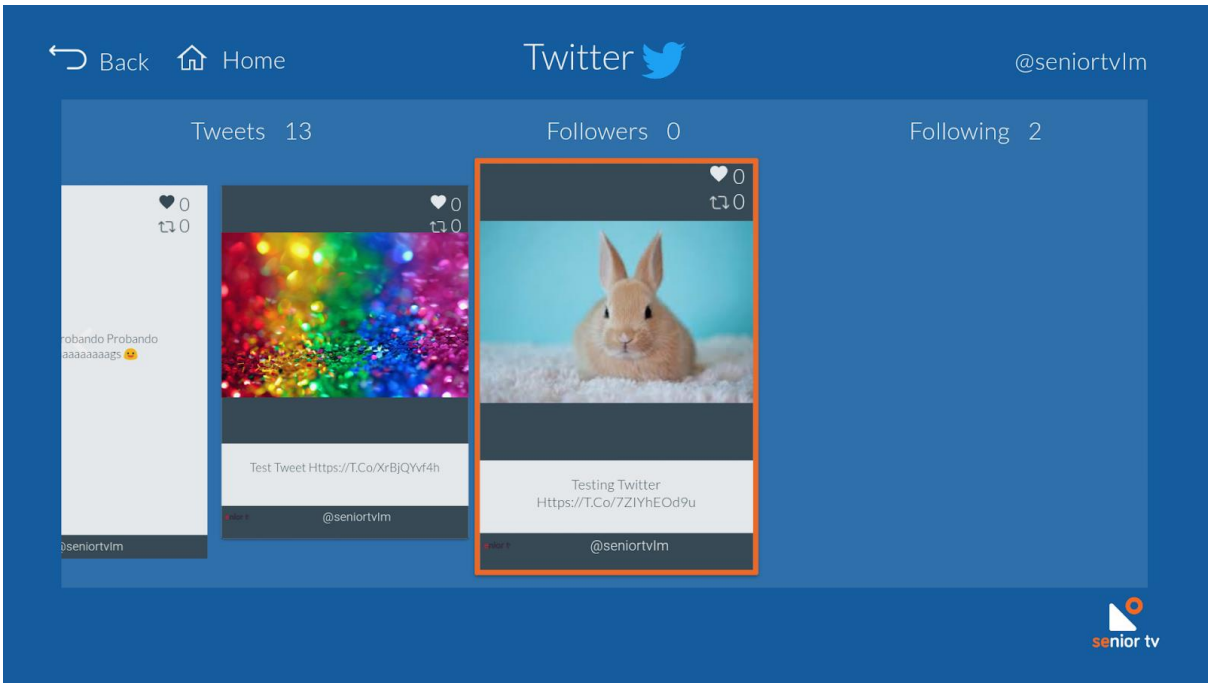


FIGURE 103. SOCIAL NETS - TWITTER POSTS

Most of the social networks include a validation process for third party apps which are going to use their data. This process is out of scope of this project and to avoid permission problems during the pilot the technical team have created testing accounts for each pilot entity. Therefore, all seniors of the same country will use the same account instead of their own accounts.

5.3.7. Rehabilitation Games

Maintaining an active life from the cognitive point of view is very beneficial for the elderly to keep a good cognitive reserve that serves as a protective cushion against diseases of a cognitive nature. SENIOR-TV platform incorporates some services which objective is stimulate and training the seniors cognitive skills. These services are presented like serious games with a dual purpose:

- To enjoy in their free time at home with motivating and leisure applications.
- To exercise their cognitive abilities in a healthy way.

The rehabilitation games included in the platform are: Corsi, Blic Block, Connect 4 and Matching pairs. The following sections include a detailed description of all of them.

5.3.7.1. Corsi 1.0

The SENIOR-TV Corsi game consists in the gamification of the cognitive evaluation test "*Corsi block-tapping test*" (Corsi cubes test), widely used for the evaluation of visuospatial memory and short-term

memory. There is computerized version of this classical test[29] using touch devices as an important step toward computerized administration of neuropsychological tests.

The Corsi Block-Tapping Task[30] is widely used for the assessment of visuospatial and short-term memory, both in clinical practice and in experimental research settings. Particularly, this test consisted of nine black cubes (30 × 30 × 30 mm) mounted on a black-colored board (225 × 205 mm). The digits 1 to 9 were printed on one side of the cubes, visible to the examiner only.

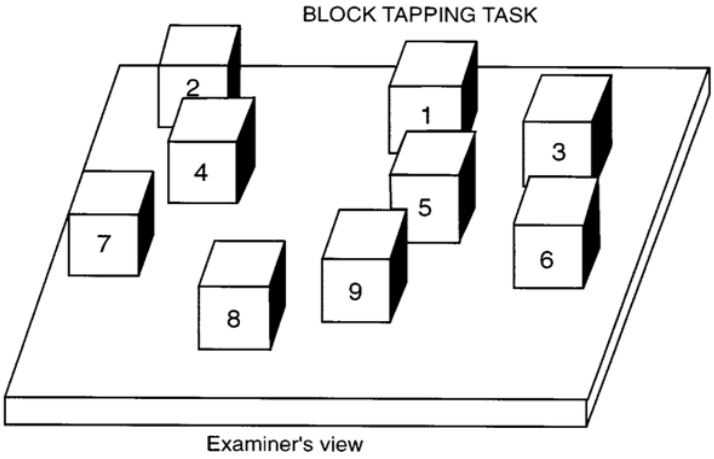


FIGURE 104. CORSI BLOCK-TAPPING TASK

The Corsi test administration follows the procedure below:

1. The participant was seated in front of the examiner, who subsequently tapped the cubes starting with a sequence of two blocks. Two trials were given per block sequence of the same length. If at least one of these was repeated correctly, the next two trials of a sequence of an increased length were administered.
2. The cubes were touched with the index finger at a rate of approximately 1 cube per sec (with no pauses between the individual cubes). The participant had to tap the cube sequences in the same order immediately after the examiner was finished. The following instruction was given: "*I will tap a block sequence on this board. When I have finished tapping, I want you to tap these blocks in the same serial order. After this, I will tap another sequence. The sequences will gradually increase in length*". If a participant started the task while the examiner was still busy tapping blocks, the following instruction was given: "*Please, wait until I have finished*".
3. The test was finished if the participant failed to reproduce two sequences of equal length. Only a completely correctly repeated sequence was scored as correct; self-corrections were permitted here.

[29] Corsi PM (1973) *Human memory and the medial temporal region of the brain*. ProQuest Information & Learning
[30] Claessen MHG, Van Der Ham IJM, Van Zandvoort MJE (2015) *Computerization of the standard Corsi block-tapping task affects its underlying cognitive concepts: a pilot study*. Appl Neuropsychol Adult 22:180–188

4. Two main **scores** are recollected:
   - Block Span: it equals the length of the last correctly repeated sequence.
   - Total Score: it is the product of the Block Span and the number correctly repeated sequences until the test was discontinued (i.e. the number of correct trials).

   According to Lezak et al.[31] criteria, a performance level of more than 1.3 SD below the control mean is regarded as "borderline," and a performance level of more than 2 SD below the control mean is classified as "retarded".

The next lists of numbers show "block sequences" examples:

8 - 5
6 - 4
4 - 7 - 2
8 - 1 - 5
3 - 4 - 1 - 7
6 - 1 - 5 - 8
5 - 2 - 1 - 8 - 6
4 - 2 - 7 - 3 - 1
3 - 9 - 2 - 4 - 8 - 7
3 - 7 - 8 - 2 - 9 - 4
5 - 9 - 1 - 7 - 4 - 2 - 8
5 - 7 - 9 - 2 - 8 - 4 - 6
5 - 8 - 1 - 9 - 2 - 6 - 4 - 7
5 - 9 - 3 - 6 - 7 - 2 - 4 - 3
5 - 3 - 8 - 7 - 1 - 2 - 4 - 6 - 9
4 - 2 - 6 - 8 - 1 - 7 - 9 - 3 - 5

The SENIOR-TV Corsi application has been implemented using Angular framework and Apache Cordova. Below are the main functionalities of the application and the main interface screens.

1. Start and play to begin playing the game.
2. Pause/resume to pause and resume the game.
3. Quit game to close the game.

[31] Lezak MD (1995) *Neuropsychological assessment, ilustrada*. Oxford university press.

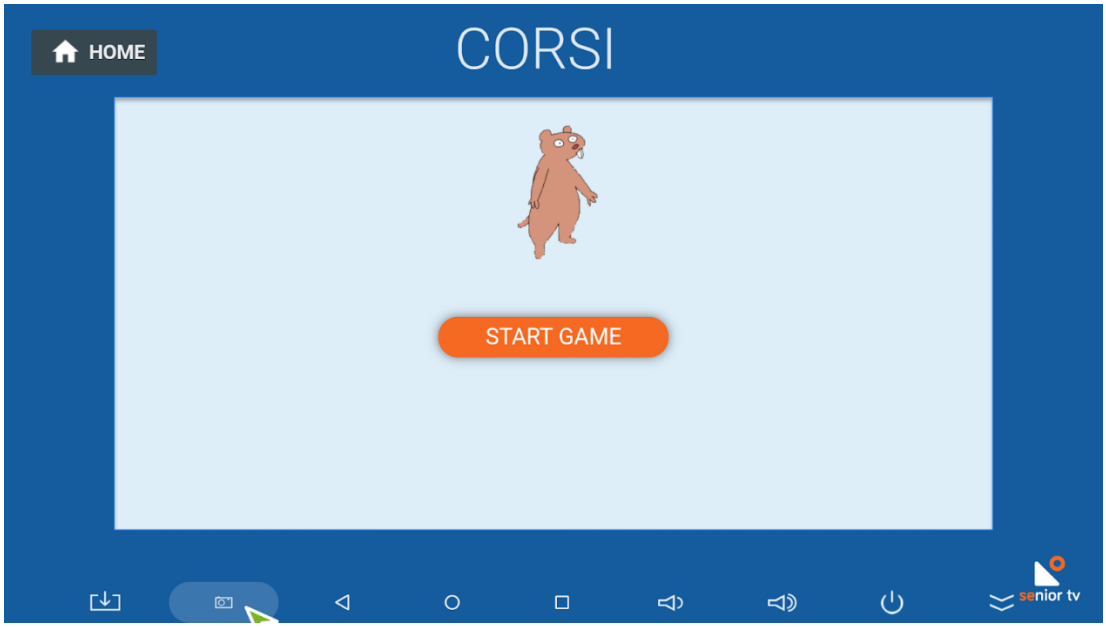


FIGURE 105. CORSI - HOME SCREEN

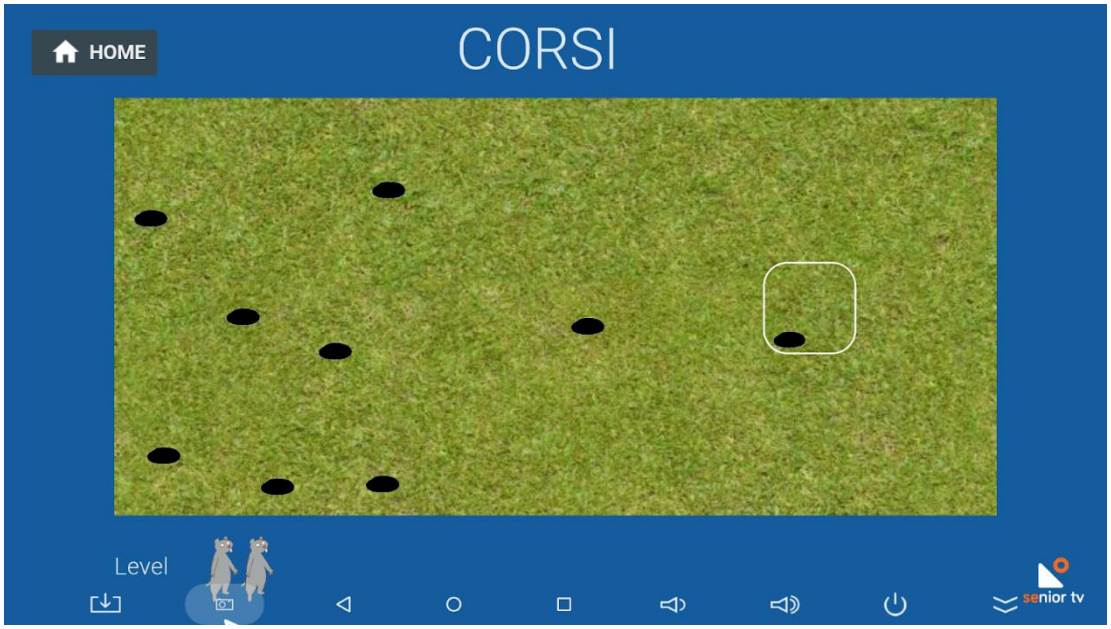


FIGURE 106. CORSI - GAME SCREEN

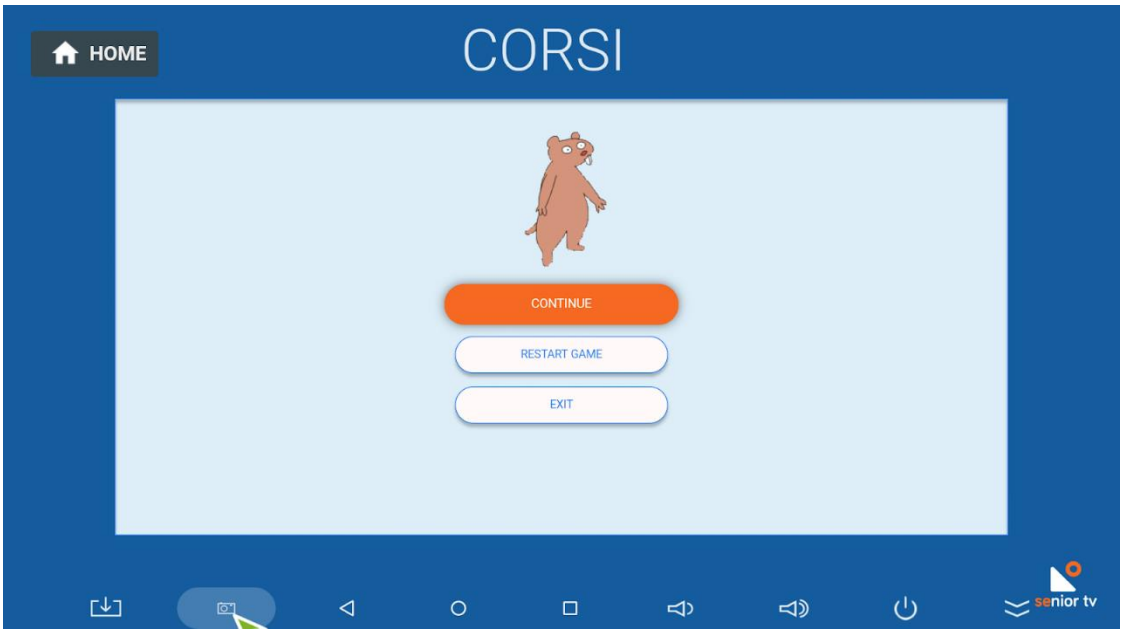


FIGURE 107. CORSI - PAUSE SCREEN

5.3.7.2. Blic Block 1.0

The Blic Block[32] game is a block game like a simplified Tetris. With this game, the seniors will train their visual perception and planification skills.

The colored squares fall from the top of the playing area and the objective is grouping four or more blocks with the same color. When this happens, the blocks disappear from the playing area and some points are received.

This TV game has been developed using JavaScript and Apache Cordova. Below, the main functionalities and an interface screenshot.

1. Start and play to begin playing the game.
2. Pause/resume to pause and resume the game.
3. Quit game to close the game.

[32] BlicBlock online game: http://blicblockjs.herokuapp.com/

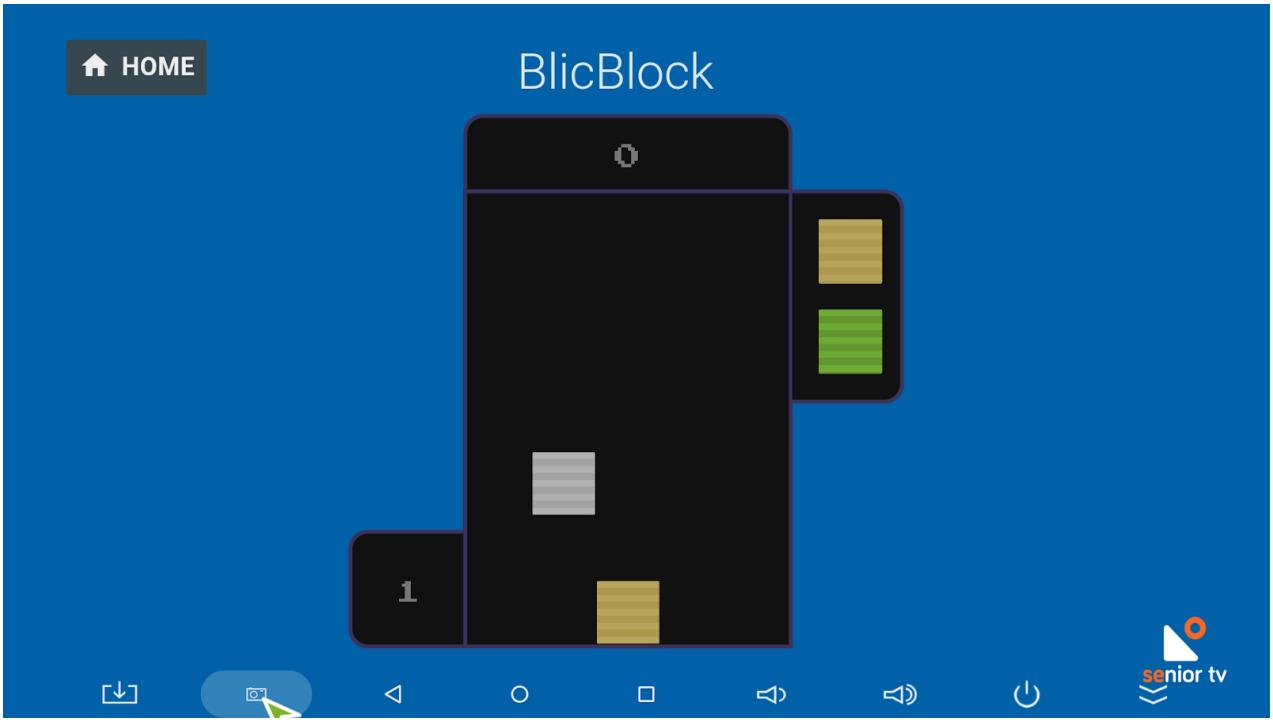


FIGURE 108. BLIC BLOCK - MAIN SCREEN

5.3.7.3. Connect 4 1.0

The Connect 4[33] game allows seniors to train their planification, visual scanning, focused attention and inhibition abilities. The game objective is to align four tiles of the same color on a board consisting of six rows and seven columns. In turns, players enter a chip in the column they prefer and it will fall into the lowest position. The first player who manages to align four consecutive chips of the same color in horizontal, vertical or diagonal direction will be the winner. If all the positions are full but nobody has made a valid row, there is a tie.

This TV game has been developed using JavaScript and Apache Cordova. Below the main functionalities and an interface screenshot.

1. Play against the computer: to start playing against the computer.
2. Play against another player: allows two users compete with each other using the same TV by exchanging the remote control.
3. Pause/resume to pause and resume the game.
4. Quit game to close the game.

[33] Connect 4 online game: https://kenrick95.github.io/c4/

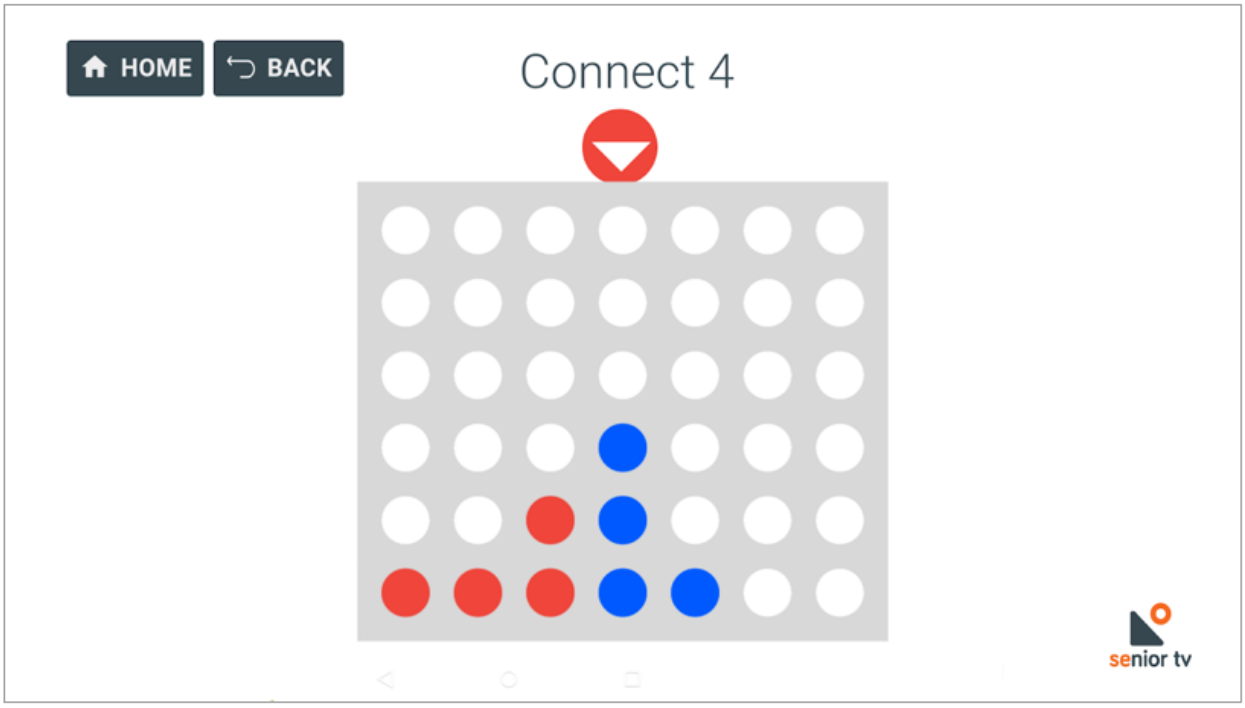


FIGURE 109. CONNECT 4 - MAIN SCREEN

5.3.7.4. Matching Pairs 1.0

Matching Pairs[34] shows a group of cards with hidden figures in each of them and the objective is finding the pairs with the same figure. This game trains the planification, visual scanning and memory of seniors.

This TV game has been developed using JavaScript and Apache Cordova. Below the main functionalities and an interface screenshot.

1. Select level: to select the game difficulty between four options: easy (6 cards), medium (12 cards), hard (20 cards) and very hard (30 cards).
2. Start and play to begin playing the game.
3. Pause/resume to pause and resume the game.
4. Quit game to close the game.

[34] Matching Pairs online game: http://www.iammaximo.com/memory-game/

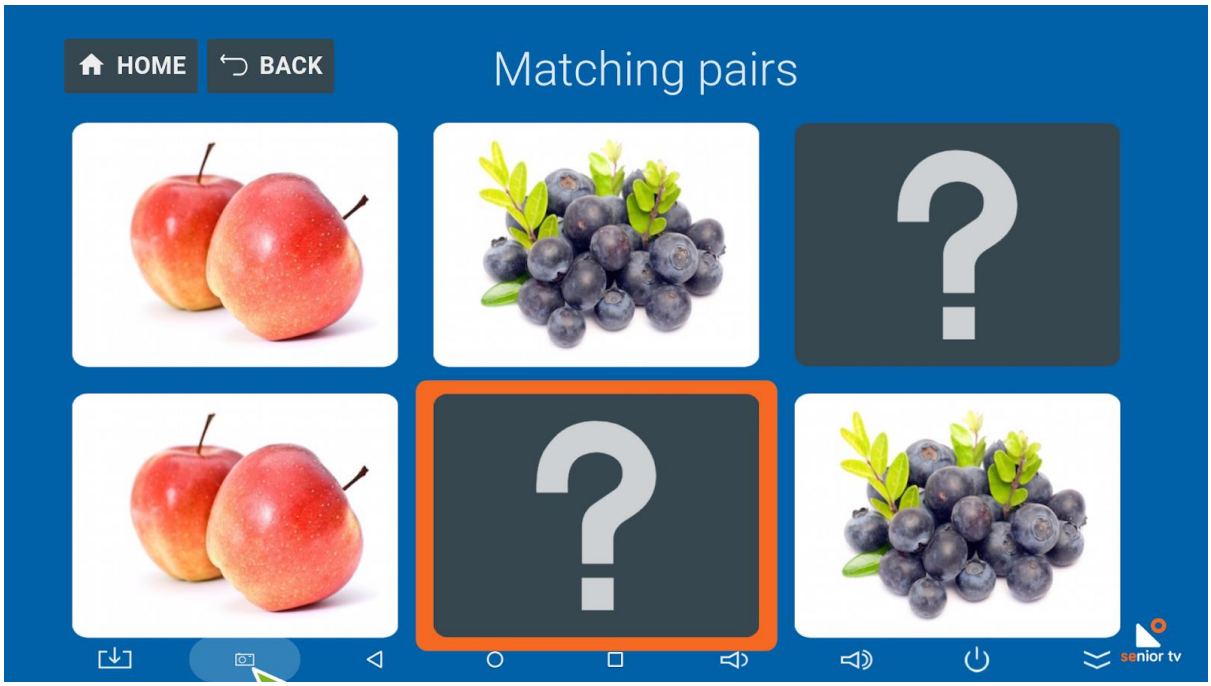


FIGURE 110. MATCHING PAIRS - MAIN SCREEN

5.3.8. Leisure games

According to the WP1 workshops' results and the Pilot 1 results, where 2 serious games were testes, games are TV apps appreciated by the seniors. Games are not only entertainment services but also help to have an active mind and encourage seniors to improve their results every day.

SENIOR-TV V3 includes three leisure games base on third-party developments. These integrations verify the openness of the platform and demonstrate that is possible to include third-party services as a part of SENIOR-TV following the usability and design guidelines agreed for the platform.

5.3.8.1. Arkanoid 1.0

The aim of Arkanoid[35] game is removing all the blocks presented at the screen hitting them with a ball. The ball bounces by all the walls of playing area except by the bottom side. At the bottom side, there is a small platform controlled by the user using the D-pad remote control, where the ball also bounces. Users have to move the platform in order to avoid that the ball will lose for the bottom side.

This TV game has been developed using JavaScript and Apache Cordova. Below the main functionalities and an interface screenshot:

1. Start and play to begin playing the game.
2. Pause/resume to pause and resume the game.

[35] Ankanoid online game: https://codeincomplete.com/games/breakout/

3. Quit game to close the game.

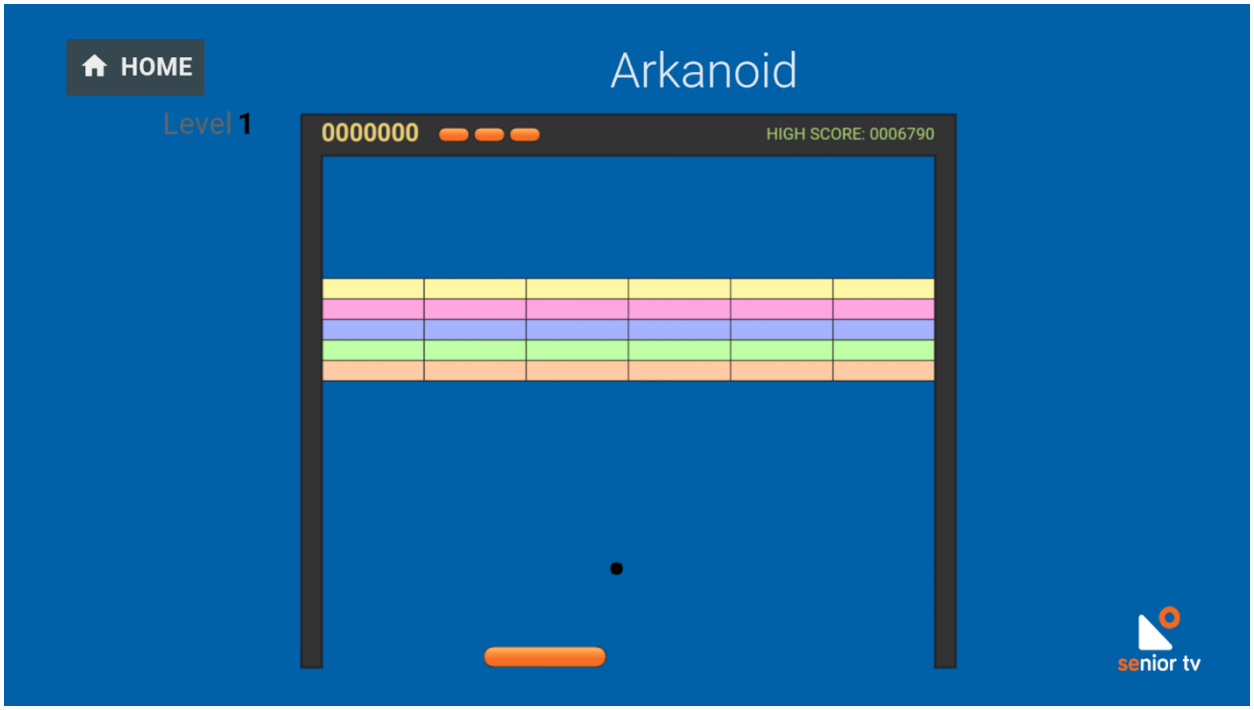


FIGURE 111. ARKANOID - MAIN SCREEN

5.3.8.2. Bubble Shooter 1.0

Bubble Shooter[36] is a logic game which objective is removing all the bubbles by aiming and hitting the right ones. Groups of two or more bubbles can be removed when they are hit by another bubble with the same colour. If there is only one bubble, the throwed bubble will be stuck next to it. At the bottom of the screen there is a trigger to shot the new coloured balls. The trigger has a pointer which indicates the shot direction and it is moved with the TV remote control.

This TV game has been developed using JavaScript and Apache Cordova. Below the main functionalities and an interface screenshot.

1. Start and play to begin playing the game.
2. Pause/resume to pause and resume the game.
3. Quit game to close the game.

[36] Bubble Shooter online game: https://github.com/rembound/Bubble-Shooter-HTML5

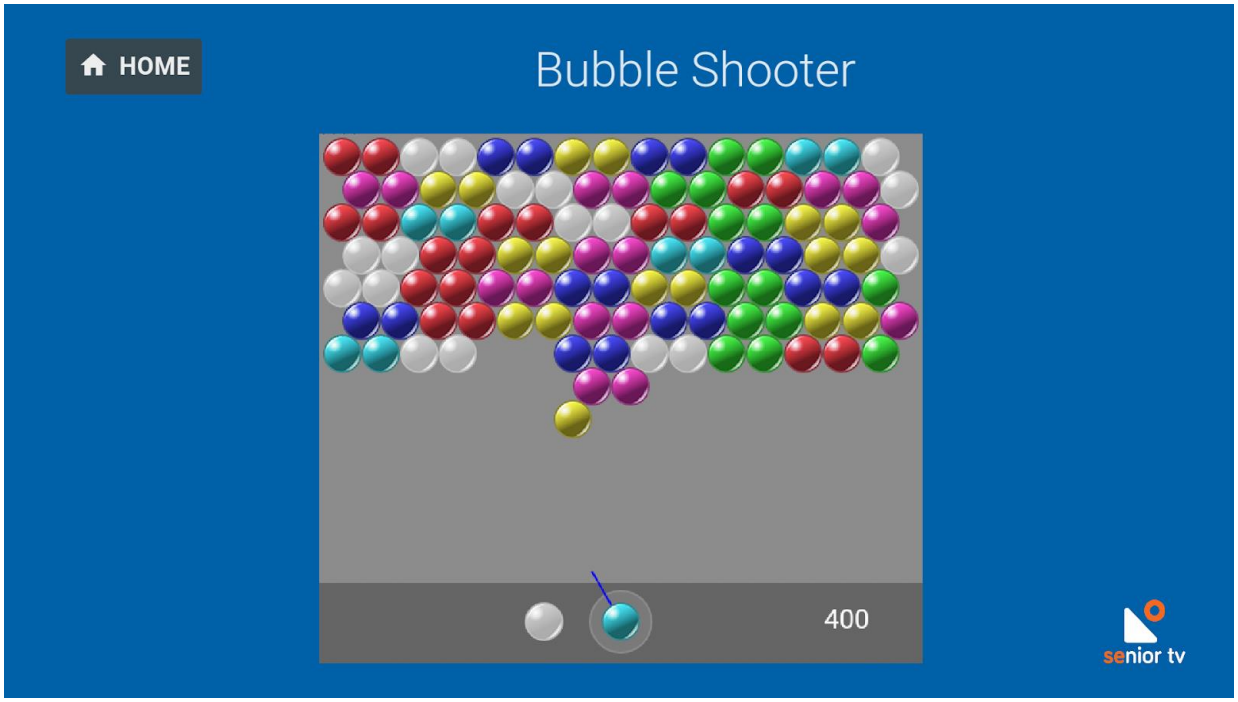


FIGURE 112. BUBBLE SHOOTER - MAIN SCREEN

5.3.8.3. Shape 1.0

The Shape[37] game objective is guessing the hidden figure in the shortest time. Figures are appearing slowly on the middle of the screen. The possible options are showing at the bottom of the screen such as eight buttons one for each figure. Users have to use the remote control to navigate through the buttons and to select the hidden figure.

This TV game has been developed using JavaScript and Apache Cordova. Below the main functionalities and an interface screenshot.

1. Start and play to begin playing the game.
2. Pause/resume to pause and resume the game.
3. Quit game to close the game.

[37] Shape online game: https://shapex.org/

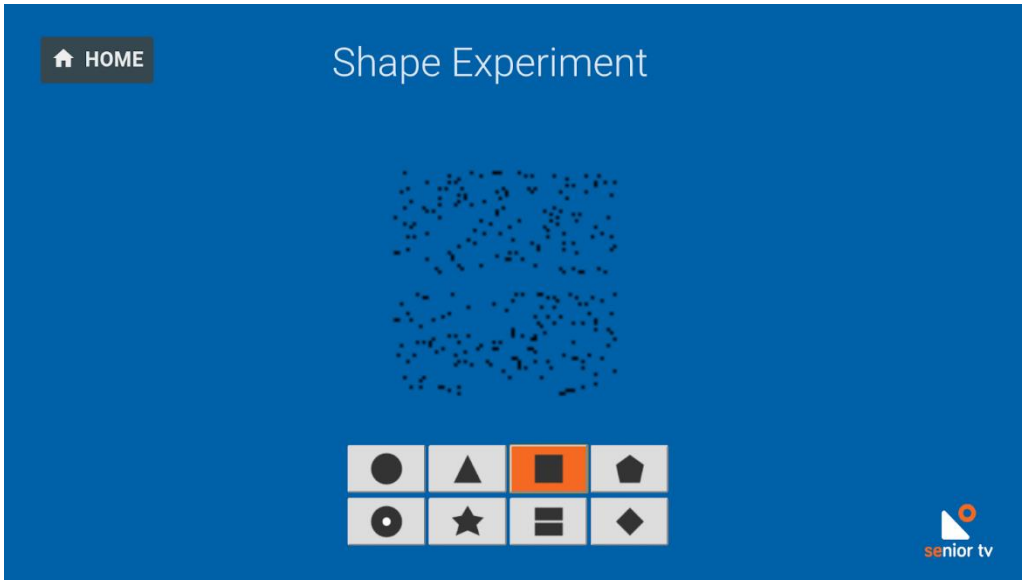


FIGURE 113. SHAPE - MAIN SCREEN

5.3.9 Out-home leisure recommender 1.0

In order to the leisure events information updated, the SENIOR-TV V3 includes the Out-home leisure recommender service. This application analyses different web sites researching the leisure activities (concerts, theatre performances, art exhibitions, village festivals, etc.) published in them, and saves the events information in a database with a standardized format. Later, this information can be used by the Events app to show new leisure activities to the seniors periodically.

The recommender information is geolocated and the events are organized by categories. Therefore, it will be able to offer upcoming events close to the senior and filtering by the certain types in which the user is interested in. In addition, the events could be in local senior languages.

The Out-home leisure recommender service is composed by two applications: cloud backend and web application.

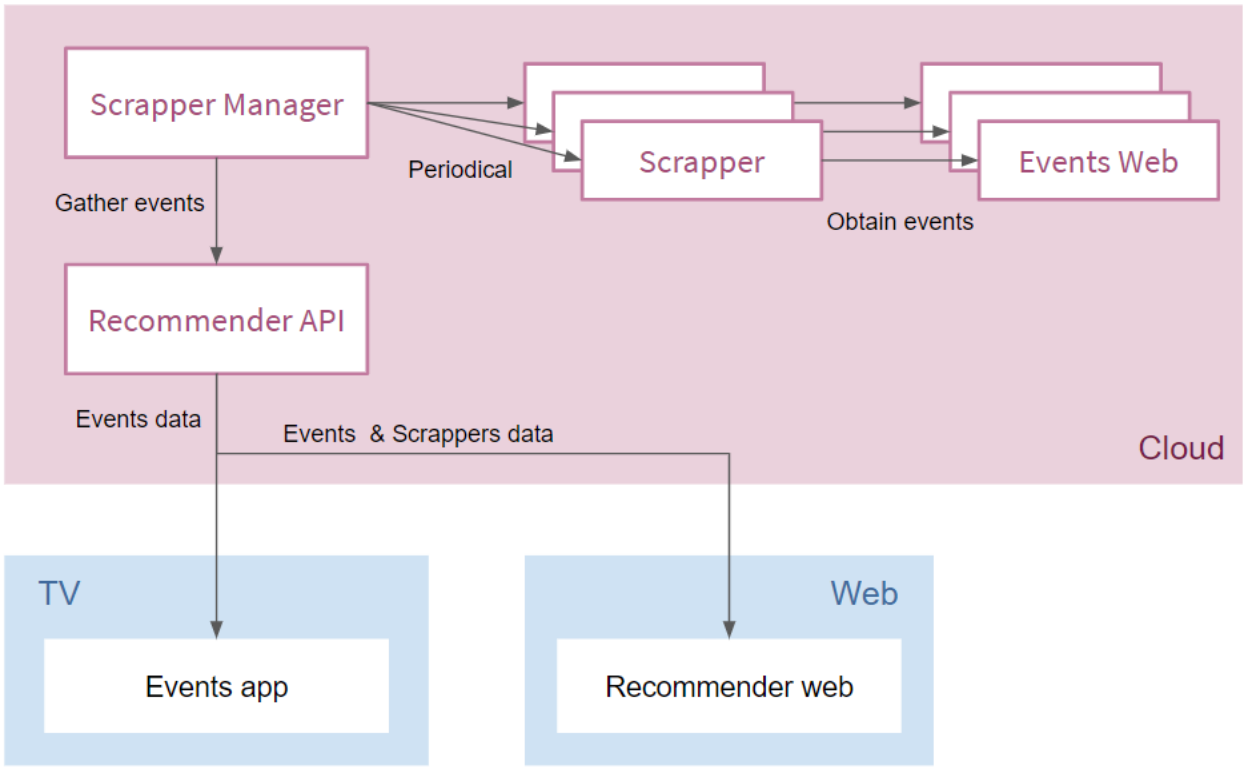


FIGURE 114. RECOMMENDER - ARCHITECTURE

The backend was developed using JavaEE/Spring and Ruby technologies and, the web app has been implemented using Angular. The next paragraphs include the functionalities details of both of them.

5.3.9.1 Recommender backend

The Out-home leisure recommender main task is to obtain, maintain, and manage the events and leisure activities that will be offered to the seniors.

1. Fetch events (web scraping): is the responsible of getting the events from the web pages. The information of the events will be automatically extracted from online sources identified by SENIOR-TV partners. This information will may be obtained from structured sources (such as RSS or ATOM) or from web pages with semi-structured information, and it will be extracted using web scraping techniques. Web scrapers are software applications able to scan websites and automatically extract relevant information from them. Notice that since each events' source website has different characteristics, it is necessary to develop specific scrapers for each identified source.

   As the Out-home leisure recommender will show events that will take place in the geographical and temporal proximity of the user, online sources identified should contain events with at least the following information: venue (GPS coordinates or address), start date, end date, description and image.

2. Manage events: include all the functionalities to store, remove and update the activities data. The information is stored in a data base where the outdated events are periodically removed. In addition, all the web scrapes are run periodically in order to get fresh events and keep the system information updated.
3. Events API is in charge of making the events available to other third-party applications. The third-party applications will be able to get the events using different filters:
    - List all available events.
    - Get detailed information of a particular event.
    - List events in a specific language.
    - List events close to the user (based on his/her GPS coordinates).
    - List events before a date.
    - List events after a date.
    - List events in a range of dates.
    - Publish a new event.
    - Update an event information.

5.3.9.2 Recommender Web

The main objective of the web administration app is showing the found events and the web scrapers information. The web application is available at https://www.imatia.com/seniortv-recommender-web and includes the following functionalities:

1. Manage events allows administrator user to see and remove the database events. The activities can be sorted by name, start date, end date or language and filtered by web source.
2. See web scraper status: a web scraping is a high time-consuming process; therefore, scrapes will be run in background. This functionality allows to know the status of the background process:
    - **Running**: the web scraper is running and fetching events from the online source.
    - **Scraped**: the web scraper execution has finished successful.
    - **Error**: an error occurred during the web scraping execution.
3. Re-geolocate events: events will be automatically geolocated based on the information obtained from the online source. This functionality allows the administrator to manually execute the geolocation functionality in case there are elements not geolocated yet.

6. Formal services

The following subsections show the main characteristics and the formal services state after the three development cycles (months 1 to 35).

6.1. SENIOR-TV V1

During the first cycle of the development of the formal services we analysed the potential possibilities for developing formal apps for the SENIOR-TV platform. This procedure took place taking in mind many aspects, such as:

- Market status and future trends.
- End users' outcome from the first iteration.
- Extensibility of the app with existing technologies and future trends.
- Integration of sensors so as to be used as input device in the app.
- Data protection, confidentiality and ethics considerations.

During this period the following technical objectives have been identified:

Technical objective	Details
Definition of apps, data collection analysis and predictions	One of the formal bundles that we are interested to develop and consists of subcomponents is the health bundle. These consist of: a) Body Weight App b) Blood pressure app c) Pulse rate app d) Blood glucose level The initial idea is that this bundle will be implemented by using a set of sensors which are connected remotely to the STB (if finally, SENIOR-TV will use an STB). The users will have the ability to view the parameters in their screen and if they wish then by just pushing a button, they could transmit this info to e.g. their doctor. Then, the doctor through a web API could give feedback, monitor the status of the elderly, etc.
Selection of commercial of the shelf sensors for the selected applications	Selection criteria will include: • Depending on the applications sensor types Power consumption evaluation of sensors and actuators for autonomy evaluation. • Sampling frequency of sensors, accuracy and sample size. • Integration possibilities

Selection of wireless technology for connectivity of the remote devices	Selection criteria will include: • Low-power connectivity solutions. • Depending on the use case corresponding design parameters such as radio transceivers density per square meter, and wireless link distance. • Low packet error rates, and wireless capacity to fulfil sensor sampling/data rates. Mesh and star topologies with the corresponding routing protocols.
Integration of lightweight communication protocol COAP of the remote devices	Traditional Web protocols (HTTP/TCP) create heavy traffic for constrained devices, therefore lightweight Web protocols needed. Constrained Application Protocol (CoAP) is a software protocol intended to be used in very simple electronics devices that allows them to communicate interactively over the Internet.
Integration of 6LoWPAN in order the Internet Protocol to be applied even to the smallest devices.	Low-power devices with limited processing capabilities should be able to participate in the Internet of Things. The 6LoWPAN group has defined encapsulation and header compression mechanisms that allow IPv6 packets to be sent to and received from over IoT based networks. Other functionalities include: address resolution, routing considerations and protocols for mesh topologies, device, service discovery and security.
Cloud integration	System architecture and development for cloud data acquisition, storage and computation. Data stored in the cloud can be processed for developing business logics. Moreover, the data can be analysed for anomaly detection, predictions, etc. depending on the use case.
Web API integration	Web API to retrieve the data and other information from the cloud and the network provides ease of integration by other nodes or free application development on the provided data.
Analytics and prediction dashboard	Optional feature – to be implemented in case there is still remaining Budget.

6.2. SENIOR-TV V2

In order to get more seniors' feedback regarding possible formal apps to include in SENIOR-TV platforms, a Health service has been developed for second pilot. The main objective of this service is collected seniors' health data using the TV. Besides, the Health service can be used by relatives or doctors to access to seniors' data because all the information is stored in the cloud.

6.2.1. Health 1.0

The Health App is the interaction of the elderly person with his/her doctor, medical organization, etc. using his/her smart-TV. All the data collected by users through the app are transmitted and stored in the cloud. Then, the secondary user side (nurse, doctor, family, etc.) through a web interface can see the transmitted values. The next image presents the data process:

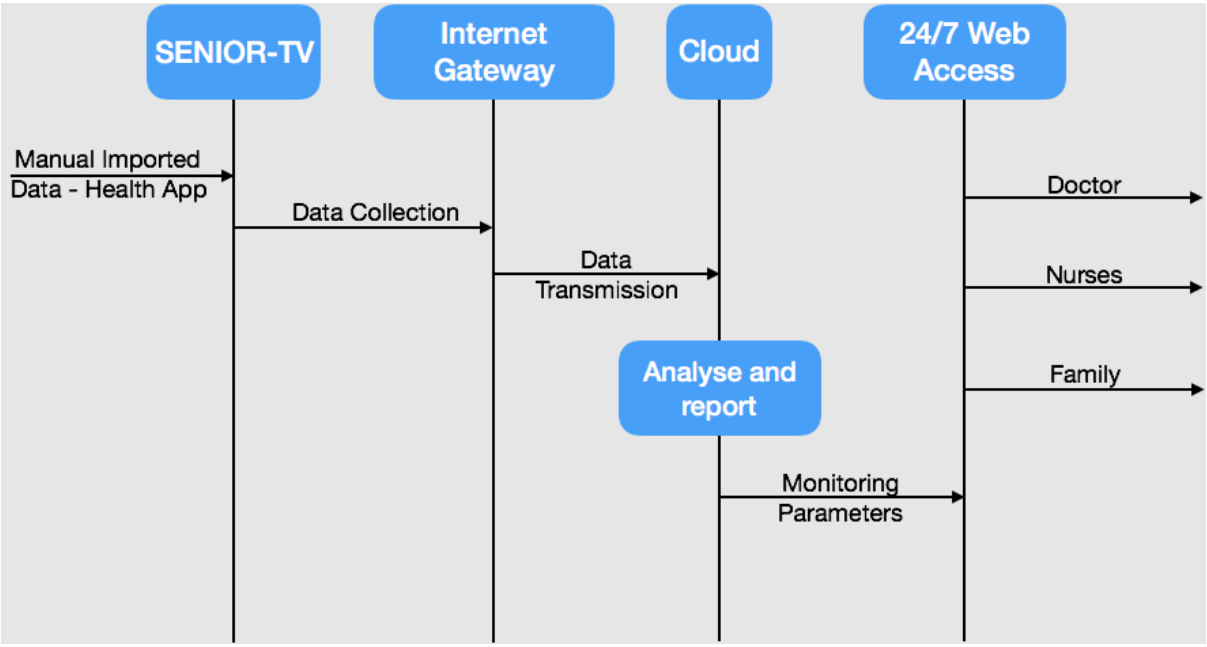


FIGURE 115. HEALTH DATA FLOW SCHEME

The app offers the following functionalities:

1. Login: via username/password screen.
2. Body weight measuring: user provides his/her blood pressure using the D-pad of the remote-control.
3. Blood pressure measuring: user provides his/her blood pressure using the D-pad of the remote-control.
4. Glucose level measuring: user provides his/her blood pressure using the D-pad of the remote-control.

5. Interface personalisation: user has the possibility to change the colours and the text size of the application.

The TV app has been developed using the Android TV SDK and is according the guidelines set to be compatible with TV devices. The initial idea during the application design was to be as simple as it could be in order to be used by the elderly. In this application the full features control is taking place using only the D-pad of the remote-control. Only authentication procedure needs to use virtual keyboard (username and password).

The deployment of the web API is taking place using Microsoft Technologies, IIS 8.5 and we use Azure cloud services. The app uses Microsoft SQL Server 2012 as for the database utilization.

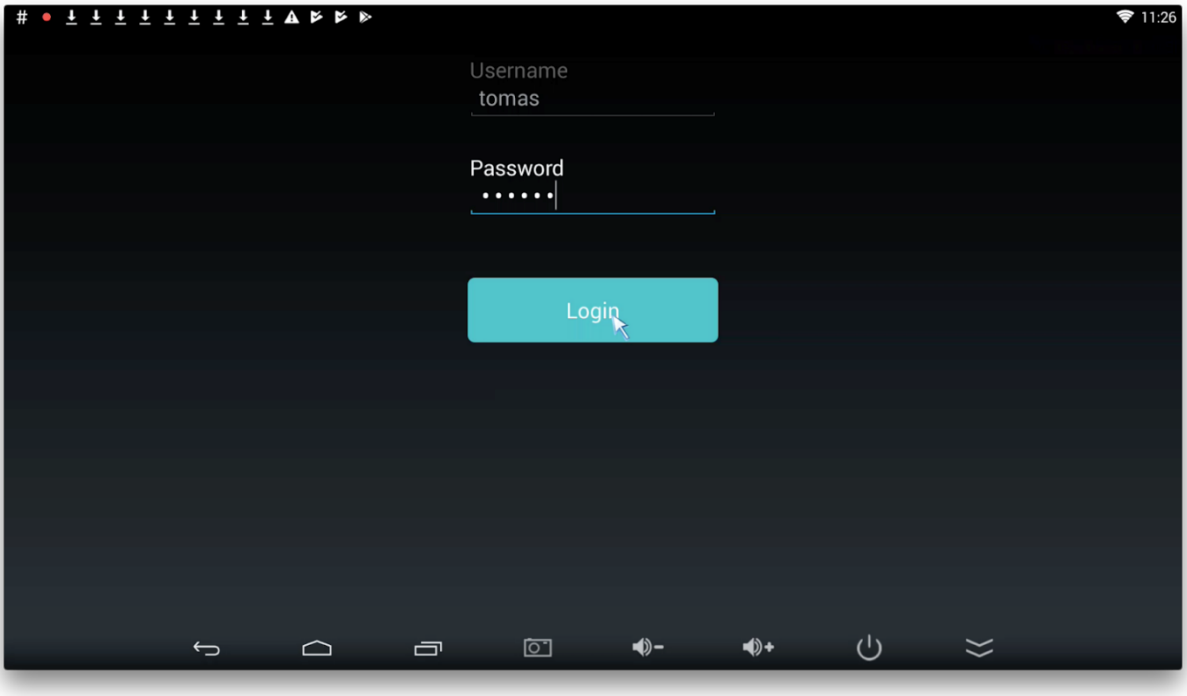


FIGURE 116. HEALTH - LOGIN SCREEN

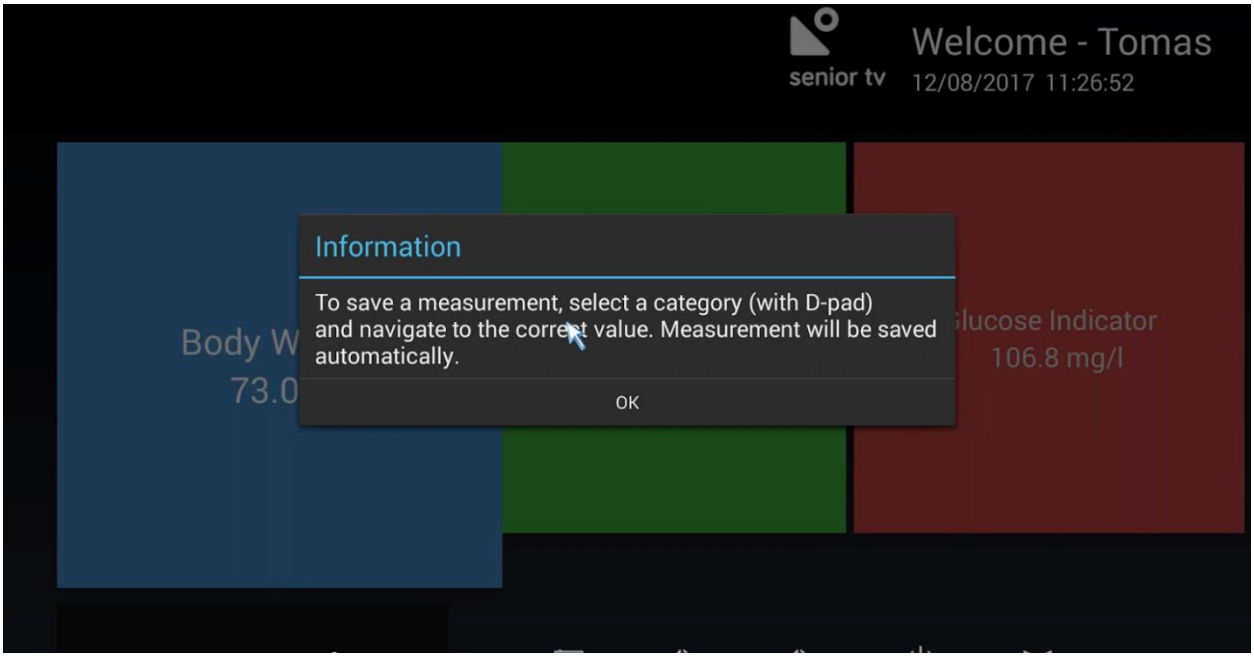


FIGURE 117. HEALTH - INITIAL SELECTION AND HELP MESSAGE

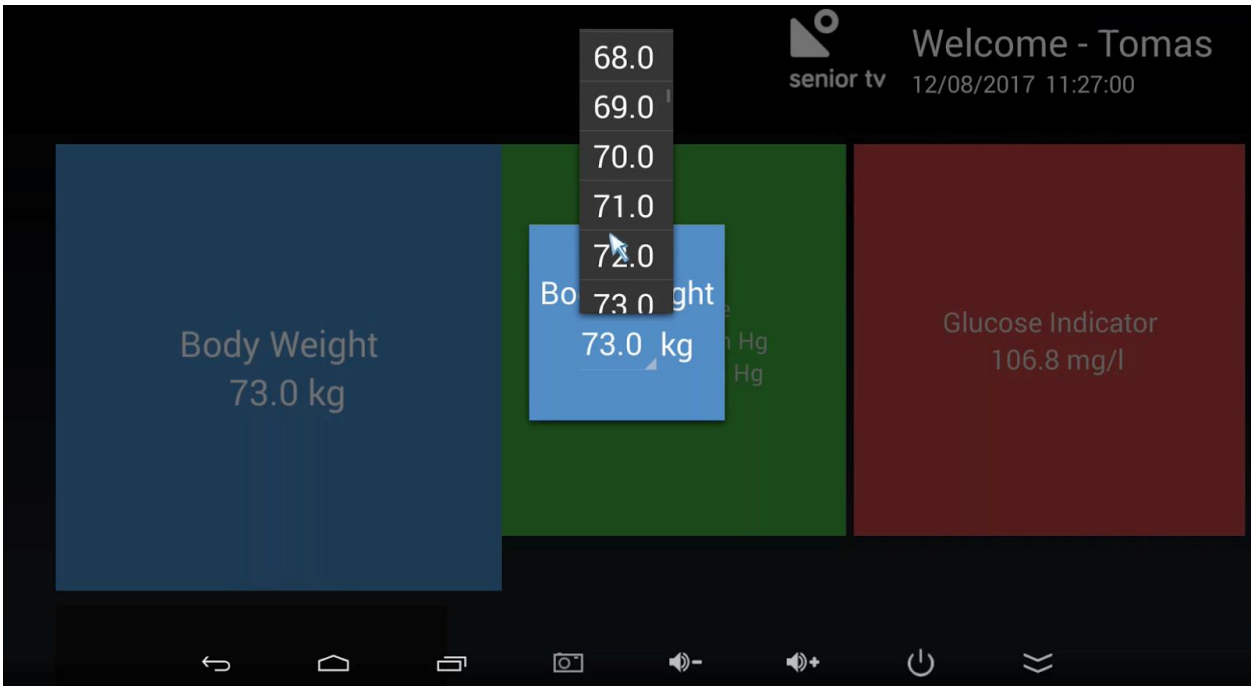


FIGURE 118. HEALTH - VALUE INDEXING

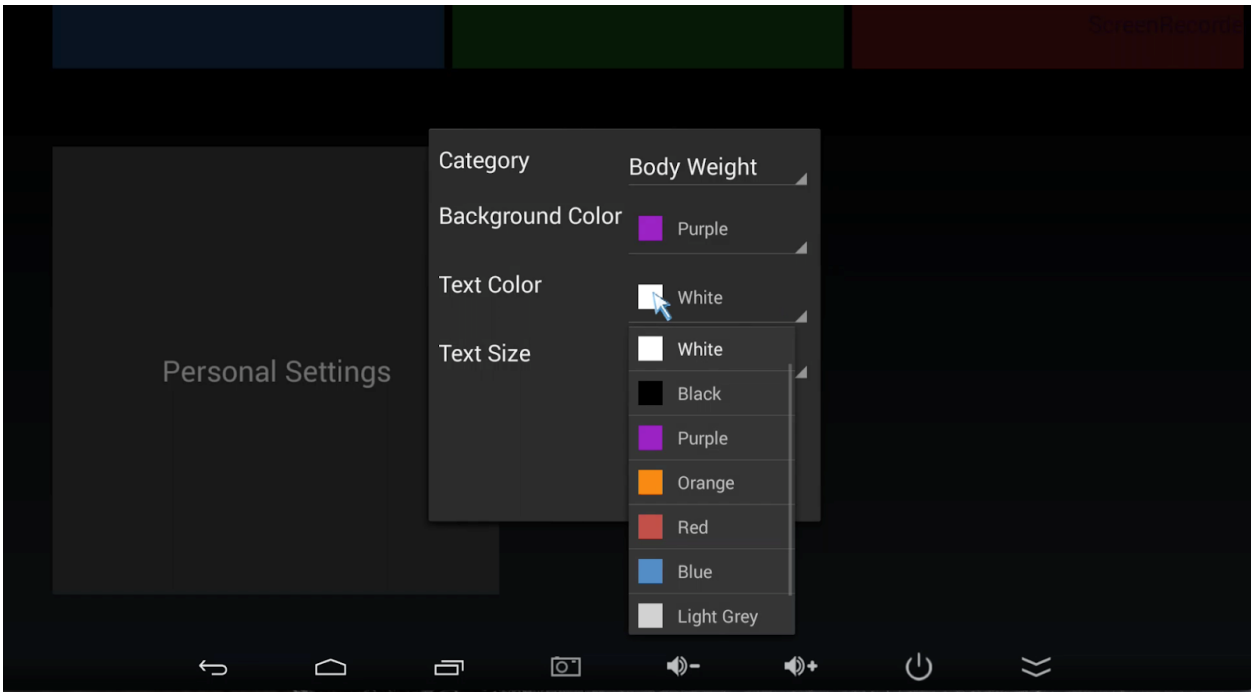


FIGURE 119. HEALTH - CUSTOMIZATION PANEL

6.3. SENIOR-TV V3

During the last annularity the formal services work has been focused on Health service interface and usability improvements, as well as integration of external sensors. The main objective is obtaining the seniors' feedback using external devices in order to get their health measurements and testing some specific sensors integration.

6.3.1. Sensor integration

In order to offer improved user experience, the functionality of retrieving automatically the body measurements using sensor devices have been developed in the Health 2.0. To include this functionality, an existing commercial product (blood pressure monitor sensor device) called Medisana® has been selected as the input device to the SENIOR-TV.

After choosing a commercial device, some lab experiments using sensor development kits and of the self-body sensors were made. However, that experimental topology was not suitable and was not possible to be tested by the end users for the following reasons:

- It was a lab testbed and cannot give the proper impression to the user of the product itself.
- Advanced engineering skills are mandatory for people that organize the trials in order to setup the testbeds.

Using Medisana® the data input takes place automatically and wirelessly using the Bluetooth® protocol. Use an existing product's wireless interface could be quite difficult since most of the products use cryptographic algorithms in order to protect their data, which makes impossible to use this information by an external-third party app.

The commercial company Medisana® gives direct access to some of their sensor devices which made possible to integrate for the blood pressure monitoring and also for the blood sugar monitoring into the Health app. For practical reasons, the sugar monitoring device has not been tested, either during the Bluetooth interface development procedure or the real life's pilots. Therefore, only the blood pressure monitoring has been integrated into the Health 2.0.

During development phase, the data from the Medisana® device has been capture directly to the SENIOR-TV box and respectively to the Health app in order to ensure that not a third-party server or cloud storage would interfere among the end user's data and the Health, and therefore preserving the data privacy.

The developed interface supports all the Medisana® Blood pressure Bluetooth® enabled monitoring devices that are using the VitaDock+ technology. Suggestive Medisana® devices that can be used, without this proposal to be tentative are the following:

- BU 540 connect Upper arm blood pressure monitor[38].
- BU 530 connect Upper arm blood pressure monitor[39].

During the market research, different models of Medisana® device were found in each EU Country. For testing the Health services the sensor device have to satisfy the following criteria:

- To be Bluetooth® enabled.
- To support the VitaDock technology.

6.3.2. Health 2.0

The major improvements comparing to the previous version can be summarized as follows:

- Interface redesign according to the common ergonomic design guidelines provided after the first and second pilots.
- The general usability has been improved according to the ergonomic design guidelines.
- Getting user data automatically using wireless sensor enabled devices.

[38] https://www.medisana.com/en/Health-control/Blood-pressure-monitor/BU-540-connect-Upper-arm-blood-pressure-monitor.html
[39] https://www.medisana.com/en/Health-control/Blood-pressure-monitor/BU-530-connect-Upper-arm-blood-pressure-monitor.html

The app functionalities have been limited at the following list, in order to reduce TV app complexity. In addition, the mandatory login process has been removed. These are the current Health app functionalities:

1. Body weight measuring: user provides his/her blood pressure using the remote control D-pad.
2. Blood pressure measuring: user provides his/her blood pressure using the remote control D-pad or taken automatically from an available Medisana®[40] device.
3. Glucose level measuring: user provides his/her blood pressure using the remote control D-pad.

Two Health app versions have been created: one for the Android TV environment and one for Android mobile devices. Each of them supports android versions as from 4.3 and above and have been developed using the Android SDK. In the House software (parsers) have been implemented to capture the data from the Medisana® device and integrate them into the app. The following sections describe the main functionalities of TV app:

1. Main screen

As soon the app is initiated you will see the main screen which contains the 3 categories: weight, blood pressure and glucose index.

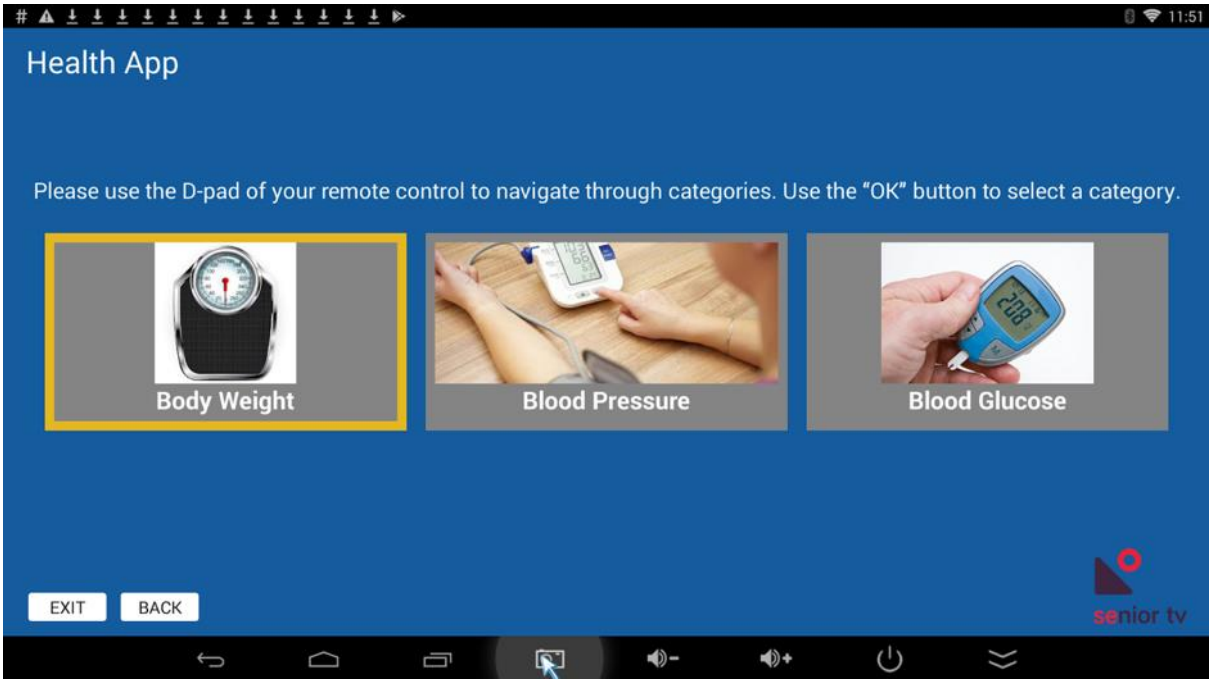


FIGURE 120. HEALTH - MAIN SCREEN

Using the D-pad, choose the desired box and then after pressing the OK key to activate it, indicate the values.

[40] Medisana® official site: https://www.medisana.com/

2. Value indexing

After selecting a measurement type, drop-out menu is shown. Use the remote control arrow buttons to choose the measurement value. Some instructions of how to indicate the measurement values appears in the TV screen in order to help to the seniors.

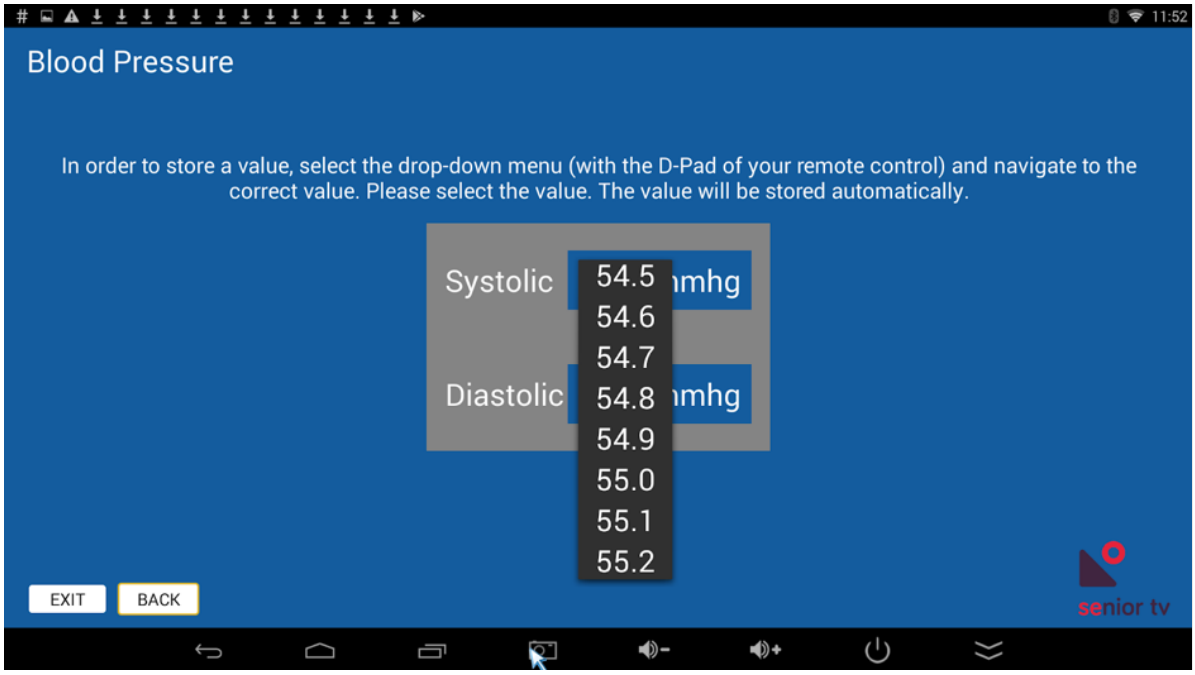


FIGURE 121. VALUE INDEXING

As soon as a value has been chosen, a notification message is shown.

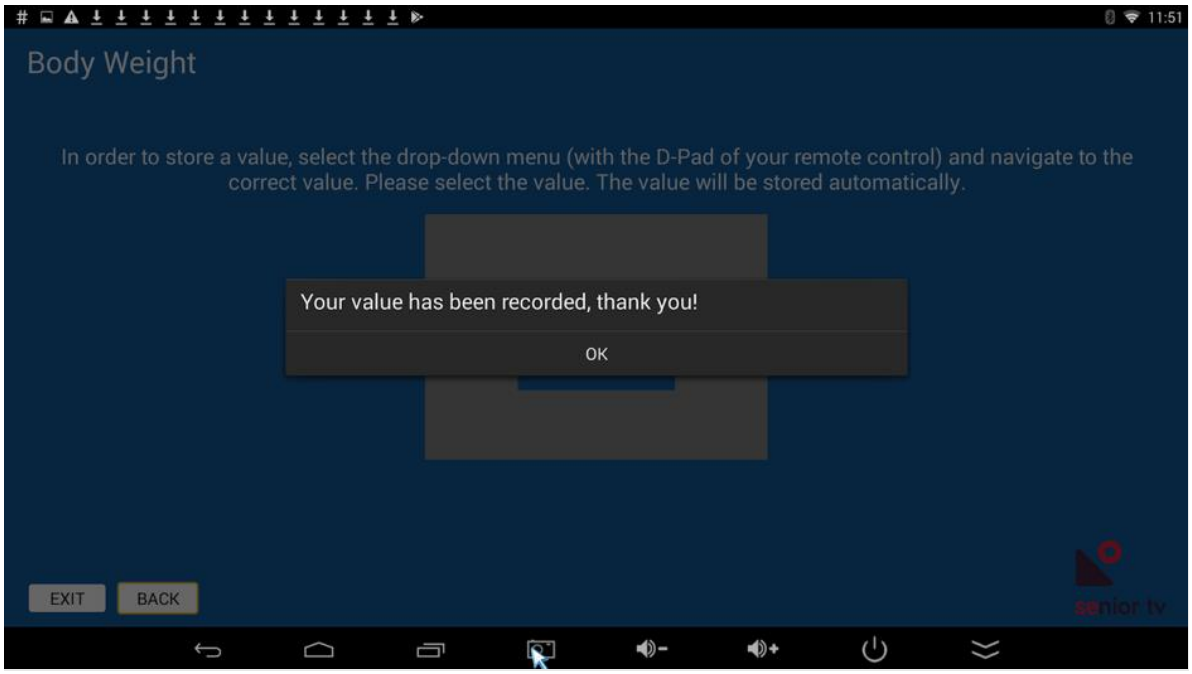


FIGURE 122. NOTIFICATION OF SUCCESSFUL ENTRY

3. Automatic mode

In the case using the Medisana® device, the next steps have to be followed to introduce the blood pressure value automatically:

1. Go to blood pressure screen.
2. Measure blood pressure on your Medisana® device or use manual sending via Bluetooth®.
3. Bluetooth® flickering icon should appear on Medisana® device.
4. Wait for 3-10 seconds, make sure that app is not going to background (the screen is not turned off).
5. See the results on the screen (a notification message is shown after receiving the Medisana® data).

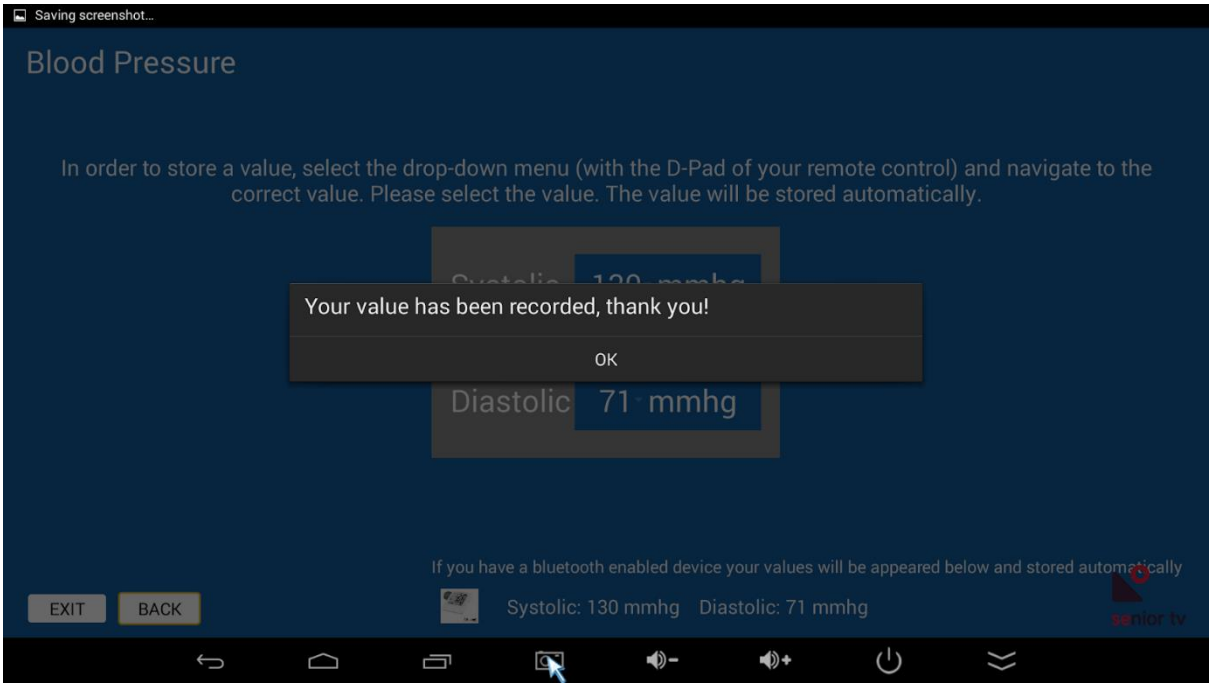


FIGURE 123. AUTOMATIC CAPTURE OF BLOOD PRESSURE

7. Ergonomic design

In this section, the TV interface design guidelines are presented. Only the Attentix and Episodix games are out of these guidelines.

While planning the design and outlook of SENIOR-TV applications, we conducted a study with University of Maribor[41] focused on what are the major points that need to be considered when building a senior friendly interface.

[41] Sasa Kuhar, Gregor Jost (2016) *Guidelines for developing TV applications for elderly. A systematic literature review.* Univerty of Maribor - Faculty of electrical Engineering and Computer science

Mostly we concentrated on following five aspects to make the solution as senior friendly as we could:

1. **Navigation**
   a. Avoid scrolling when possible, because this is something that seniors are not accustomed to and have difficulty comprehending it.
   b. Headings of different pages must be unique but they should also have a related look, so users know they are a part of the same solution.
   c. All the pages must have titles on top, so users know where they are currently positioned.
   d. Entire navigation must be simple and consistent throughout the applications.
2. **Language**
   a. Expert terms should be avoided and language used should be simple.
   b. Common words should be used and if possible, avoid newer or slang words.
3. **Text**
   a. Font used should be easy to read (simple font).
   b. It is advisable to use only one font throughout all the applications.
   c. Font must be large enough as seniors have difficulties reading smaller text.
4. **Buttons**
   a. Selected items should be clearly highlighted so it is easy to distinguish between selected and non-selected items.
   b. There should be high contrast between buttons, background color, and text color.
   c. Buttons should be big and there must be enough space between them.
5. **Colors**
   a. Bright colors must be avoided.
   b. Negative polarity is preferred over positive (light color text on dark color background is better than vice versa).

7.1. SENIOR-TV V1

Based on the University of Maribor knowledge and the SENIOR-TV logo colors the first front end guidelines were designed. Next paragraphs show that design details.

FIGURE 124. SENIOR-TV LOGO

7.1.1. Color swatches

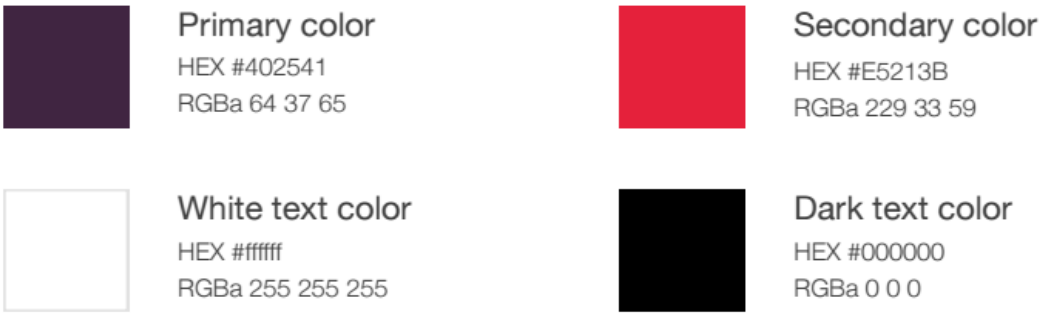


FIGURE 125. FIRST PILOT COLOR SWATCHES

7.1.2. Typography

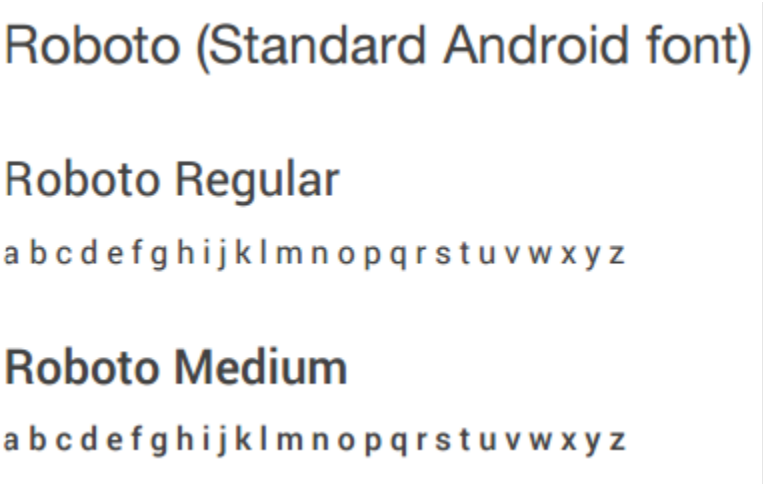


FIGURE 126. FIRST PILOT TYPOGRAPHY

7.1.3. Button states

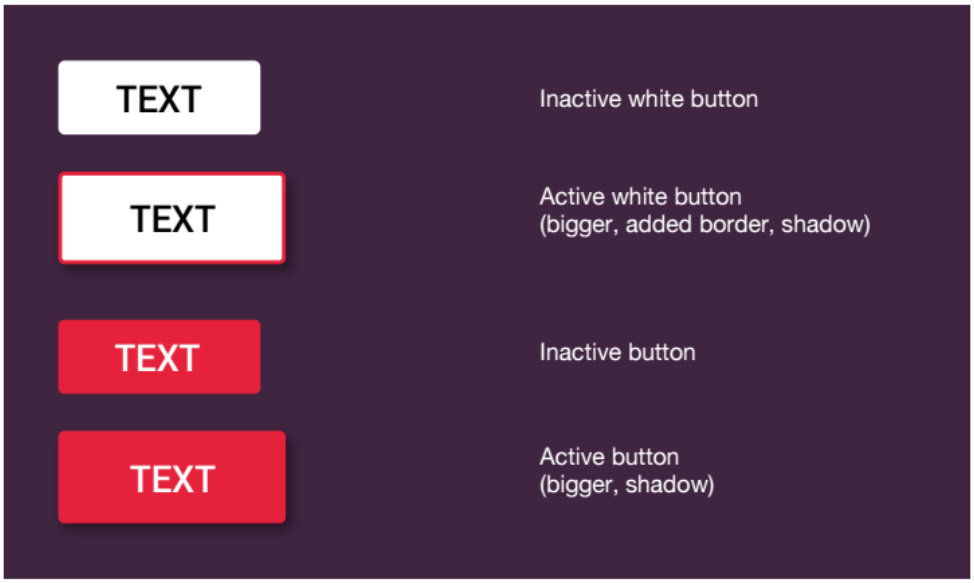


FIGURE 127. FIRST PILOT BUTTONS

7.1.4. TV interact and remote-control

Users navigate apps using a directional pad (D-pad) and using the air mouse pointer. This type of controller limits movement to up, down, left, and right.

FIGURE 128. AIR MOUSE REMOTE-CONTROLLER USED IN FIRST PILOT TESTING

7.1.5. Navigation, focus and selection

A key aspect of making application work well with a D-pad controller is to make sure that there is always an object that is obviously in focus. App must clearly indicate what object is focused, so users can easily

see what action they can take. Use scale, shadow brightness, opacity, animation or a combination of these attributes to help users see a focused object.

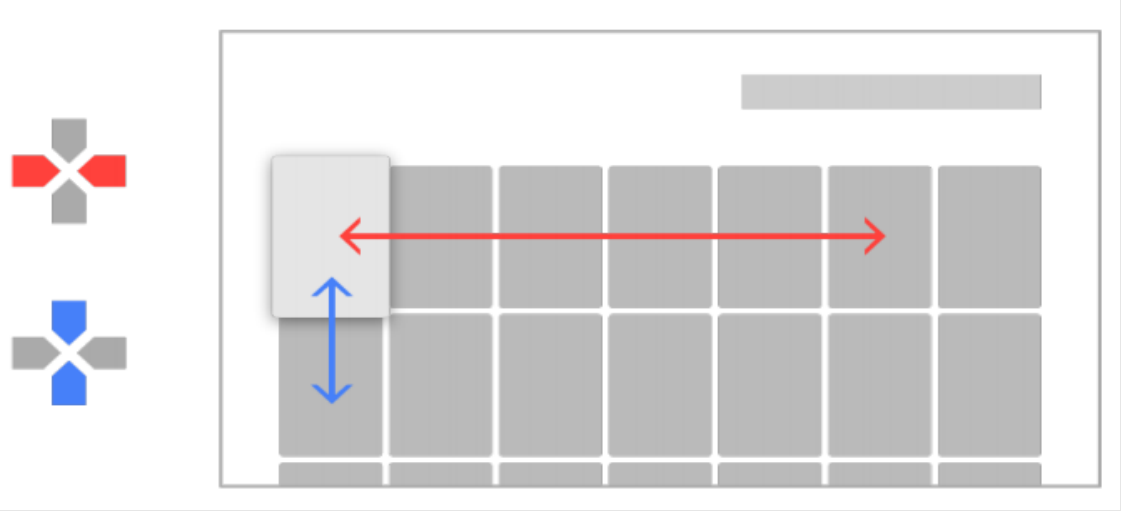


FIGURE 129. NAVIGATION

7.1.6. Senior feedback

After presenting our applications to end users during first pilot testing, the vast majority commented negatively on the colors we used, especially the background color, describing it dull and depressive. We took this feedback into account and prepared several different color schemes.

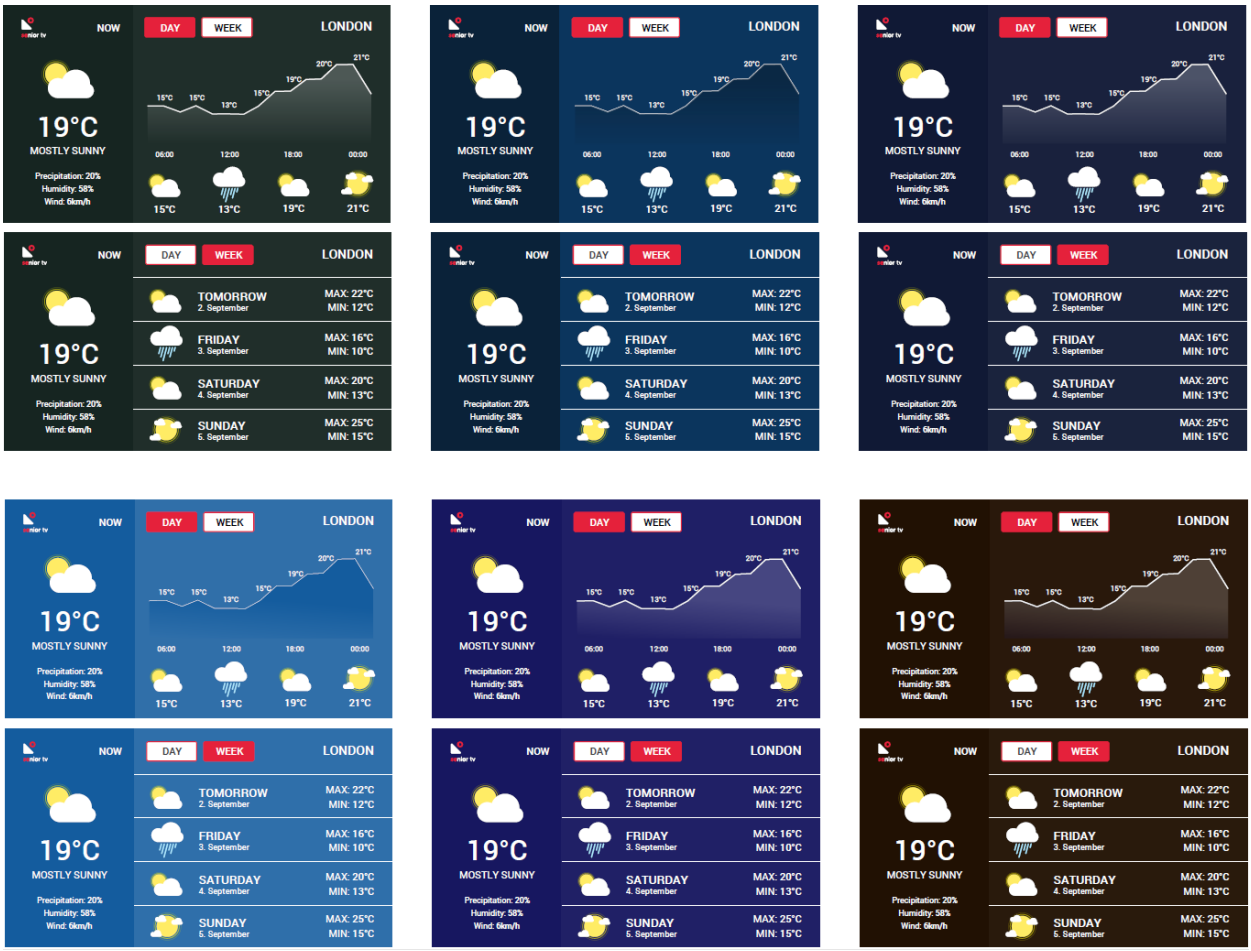


FIGURE 130. ALTERNATIVE COLOR SCHEMES

After presenting the color schemes to several groups of end users, they were all in favor of the light blue background scheme. This is why we chose the blue color scheme and implemented it into the second pilot's applications.

While conducting our first pilot testing, we noticed that it was hard for end users to control SENIOR-TV applications by using the air mouse remote-controller. They had problems keeping steady hands while pointing the device and it was hard for them to keep the remote still while pressing buttons.

7.2. SENIOR-TV V2

Considering the first pilot feedback and the results of the alternative color schemes tests, the user interface of platform second version was focused on light blue colors and the use of remote control D-pad to navigate through the screens.

7.2.1. Color swatches

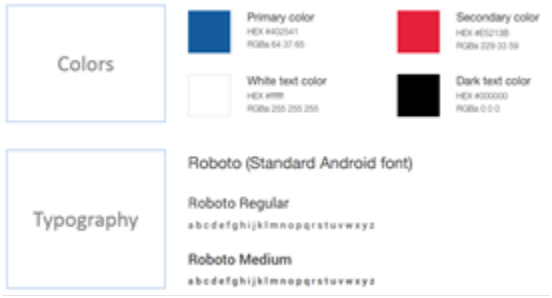


FIGURE 131. SECOND PILOT COLOR SWATCHES

7.2.2. Button states

FIGURE 132. SECOND PILOT BUTTONS

7.2.3. TV interact and remote-control

Due to seniors' problems with the air mouse, some alternative D-pad remote-controls have been studied. We used three groups of end users to test which controller was most suitable for them and the results were unanimous – they all preferred the Minix standard remote-controller with very few buttons and easy to use navigation arrows. They were accustomed to the use of such remote-controllers and their unsteady hands were no longer a problem.

FIGURE 133. D-PAD REMOTE-CONTROLLER USED IN SECOND PILOT TESTING

7.2.4. Other interface considerations

End users often had trouble using physical BACK button on our remote-control and we also got feedback while holding our midterm review meeting that both HOME and BACK buttons should be always present on every screen. This is psychologically positive because it is reassuring to seniors that they always have a way out of the application and back to home screen. This is something that makes them feel safer. We thus added HOME and BACK buttons to every screen to help end users with their navigation and to help them feel more at ease while using our applications.

The following image shows the main elements in the SENIOR-TV apps:

1. Service name.
2. Logged-in user name.
3. Back and Home buttons.
4. Platform logo.

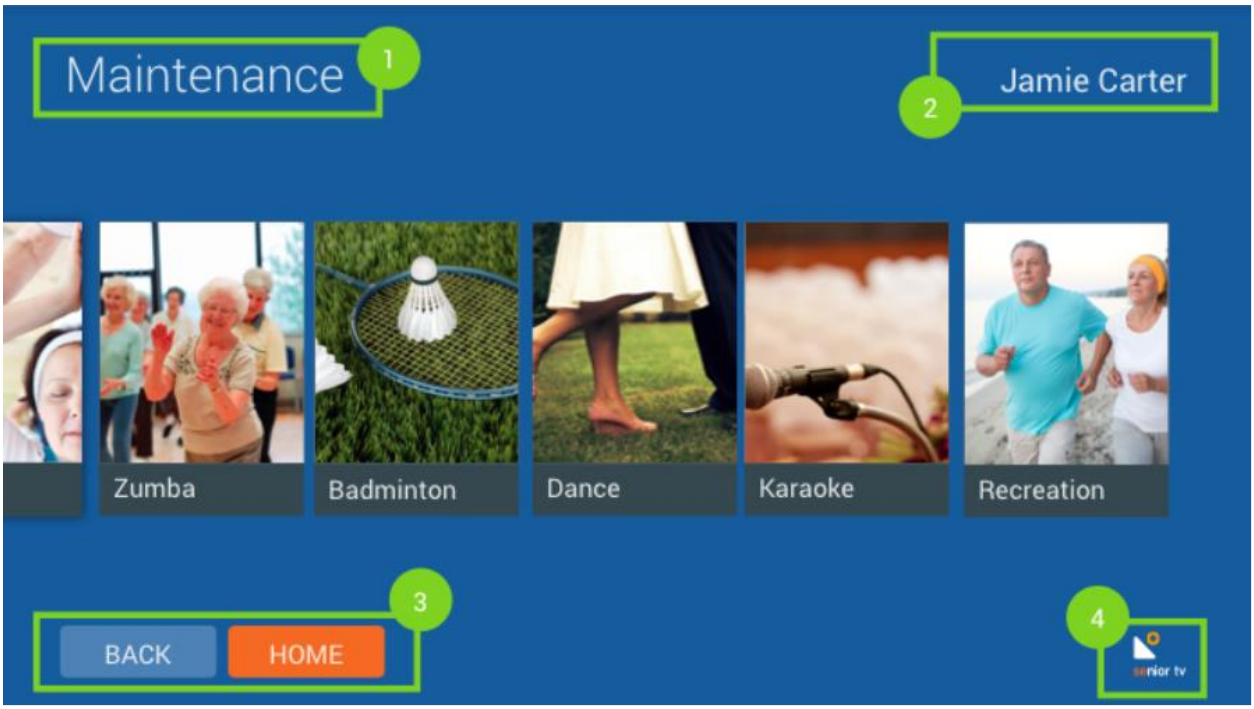


FIGURE 134. GENERAL INTERFACE LAYOUT

7.3. SENIOR-TV V3

During the final annuity the works on this section were focused on studying alternative devices of classical D-pad remote control. The interface design of the services incorporated in the SENIOR-TV V3 continued with the guide lines adopted during version 2.

The seniors' feedback got during second pilot confirmed that the light blue interface and the D-pad remote control are better options than the dark garnet color and the remote pointer used during first pilot. However, some difficulties were reported to interact with the TV app using D-pad keys. Therefore, an alternative interface has been developed based on touchable devices such as tablets and mobiles.

7.3.1. Touch Remote Control

The Touch Remote Control service allows users to use a touch device such as a smartphone or a tablet to interact with SENIOR-TV platform. The service is composed by two apps: The *Remote Control Service* for the Android TV device and, the *Remote Control application* for an Android mobile or tablet.

The following chart shows the service architecture and the details are explained below:

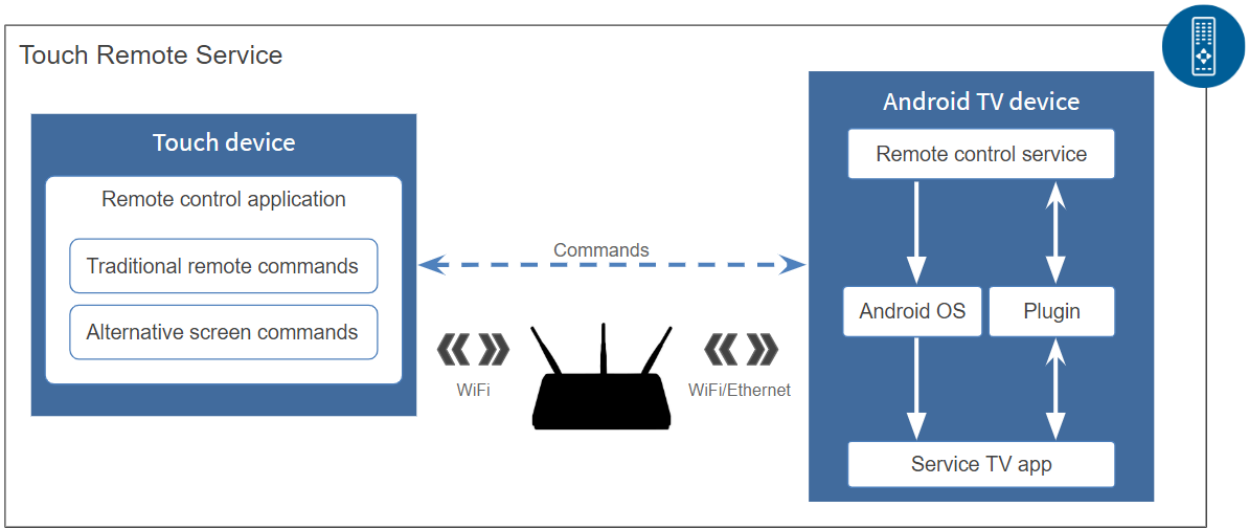


FIGURE 135. TOUCH REMOTE CONTROL ARCHITECTURE

1. Remote Control Service has to be installed in the Android TV device in order to handle the commands sent by the *Remote Control application*. This application works as an Android service, and it will start when the Android Minix device starts.
2. Remote Control Application is the mobile application to control the SENIOR-TV. It is the responsible of sending the *commands* to the *Remote control Service*, installed in the Android TV device. This app has two interfaces with different functionalities:
   - **Traditional remote**: is a clearest and simplest replica from physical Minix remote control, avoiding confusing and not necessary buttons for senior users, such as settings and change desktop buttons.

     This interface will send "*traditional commands*" to the TV Service. The traditional commands correspond with the physical remote control actions: left, right, up, down, OK, Volume Up, etc. As a result, the *Remote Control Service* will send the received commands to the Android Operating System and it will be responsible for executing the command on the TV App.
   - **Alternative screen**: is a completely different interface to interact with SENIOR-TV apps. The objective is not using the D-pad arrow concept and showing the TV app options directly in the touch device, and not use the arrow keys or the OK button any more. For now, only the Shape game (see section 5.3.8.3. Shape) is adapted to work with this alternative screen.

     This interface will send "*alternative screen commands*" which equivalent to the actions performed by the alternative screen interface, for instance, "click on the triangle" in the Shape game.

The main restriction of Touch Remote Control is both the mobile/tablet and the Android TV device, have to connect to the same Local Area Network. The mobile device will be connected via Wi-Fi, and the Android TV device will be able to be connected using Wi-Fi or Ethernet.

The following figures describe in detail both Remote Control Mobile interfaces: traditional and alternative.

1. Traditional interface

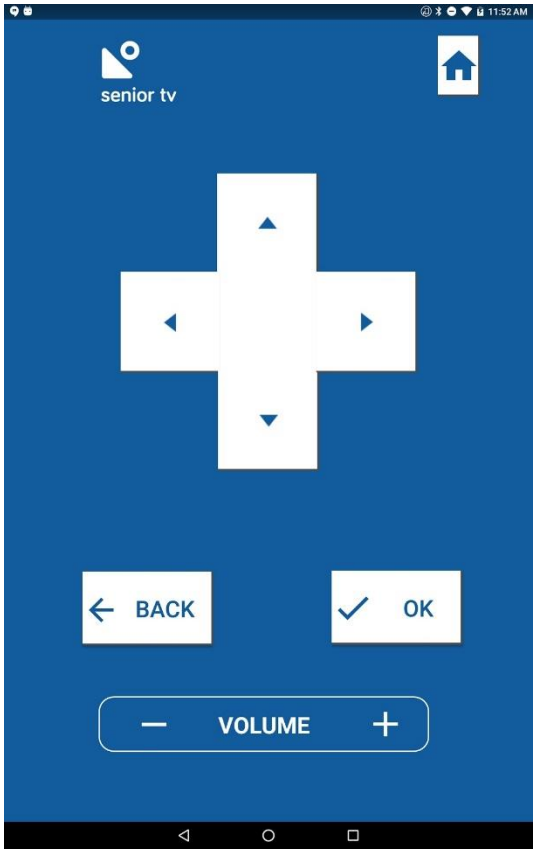


FIGURE 136. TRADITIONAL REMOTE CONTROL

2. Alternative interface

This is an alternative screen to play the Shape game in a different way, without using the arrow keys or the OK button on the remote control. If a touch *Remote Control Mobile* is connected by the Android TV, when the Shape game starts, the Alternative interface is automatically opened.

The options shown by the Alternative screen depend on the TV app. In the case of Shape game, the options shown correspond with the 8 possible figures to guess (see next figure).

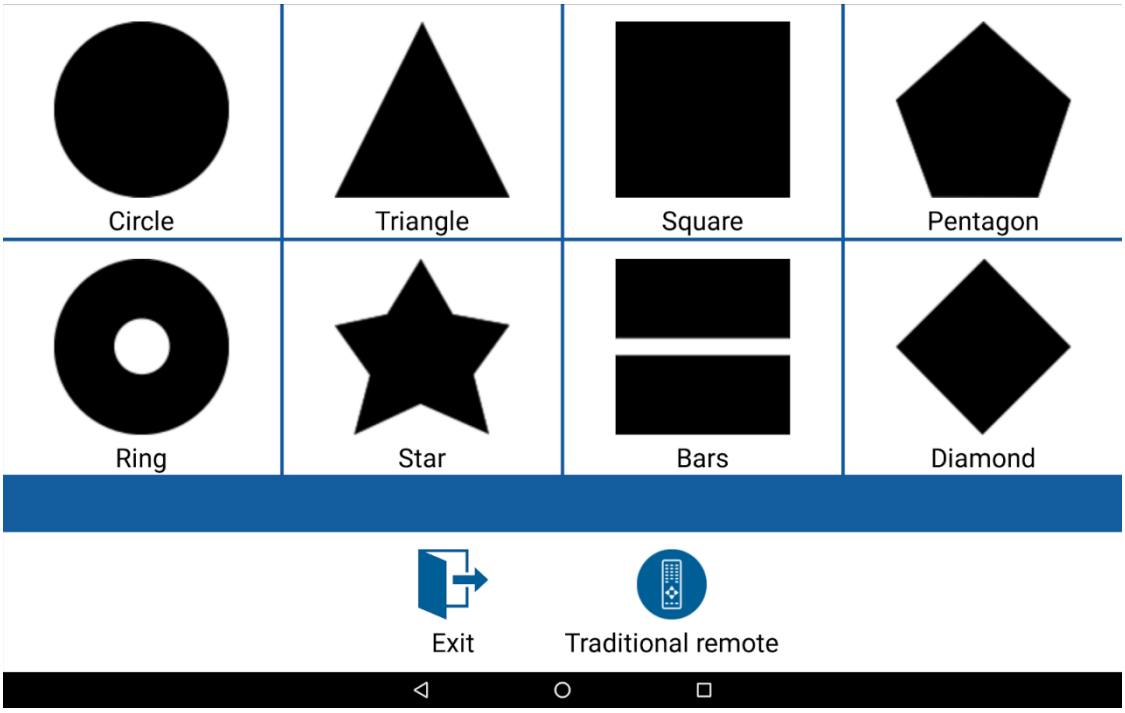


FIGURE 137. ALTERNATIVE REMOTE CONTROL - OPTIONS

Alternative touch control facilitates the interaction with the Shape game: users only have to touch on the mobile/tablet screen the figure that considers correct instead of using the arrow keys to select the desired figure and the OK button to select it.

Furthermore, seniors receive better feedback than traditional remote control. When user chooses the correct figure, the selected figure will show a green background on mobile device. In other case, the background will be red.

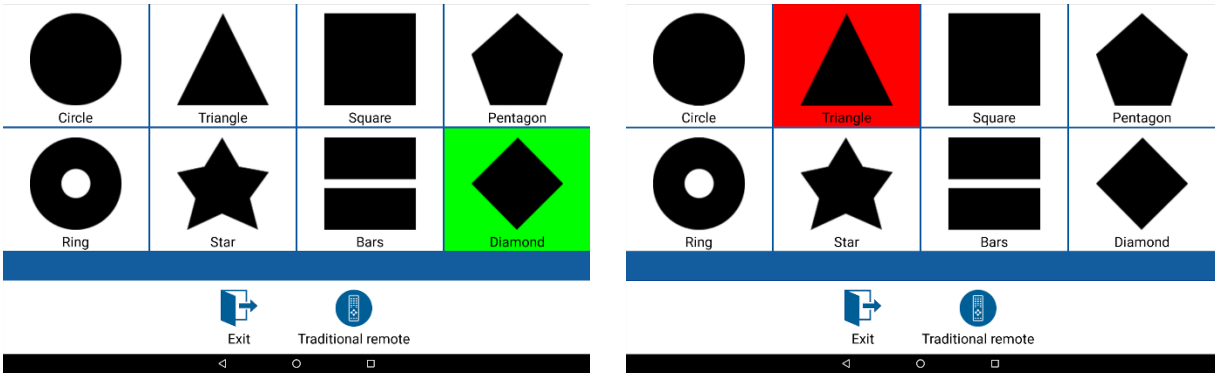


FIGURE 138. ALTERNATIVE REMOTE CONTROL – FEEDBACK COLORS

8. System Integration

All the SENIOR-TV platform are supported by a System Integration module. This module is composed by the Integration Cloud, the AuthApp and the Client TV Ecosystem.

- The Integration Cloud is the responsible of cloud backend functionalities necessary to support the SENIOR-TV platform such us user management, authentication, cloud notifications and device management. The Cloud System Integration services are consumed by the SENIOR-TV services, the AuthApp and, the Client TV Ecosystem.
- The AuthApp uses the Integration Cloud functionalities to identify the SENIOR-TV users on the TVs and mobile devices and to manage platform notifications.
- The Client Ecosystem main objective is created a compact TV system interface to access to the formal and informal services and to Integration Cloud functionalities from only one application running in an Android TV device. Ecosystem facilitates the interaction with the SENIOR-TV platform hiding no needed Android TV system characteristics.

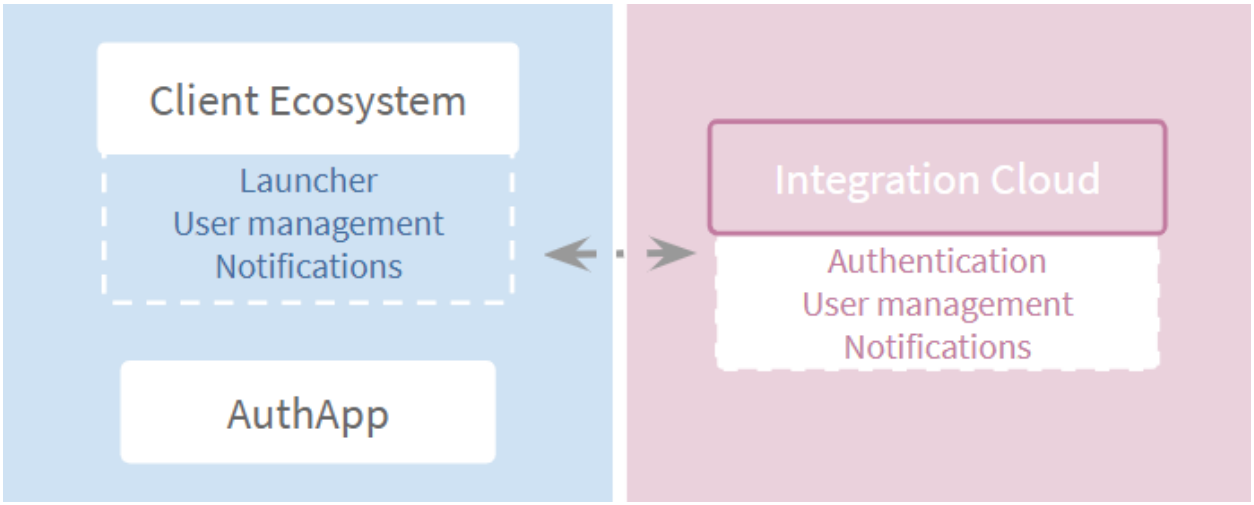


FIGURE 139. SYSTEM INTEGRATION ARCHITECTURE

8.1. Integration services

The SENIOR-TV integration services are formed by two types of applications: cloud and mobile.

- The cloud services correspond with backend applications which are responsible to save and manage users and their devices data. That information is used to manage the platform user sessions and to manage the services notifications.
- The mobile or TV services main functionalities are authenticating the users at the SENIOR-TV platform and managing the system notifications.

The following diagram shows the most important components of the Integration Services:

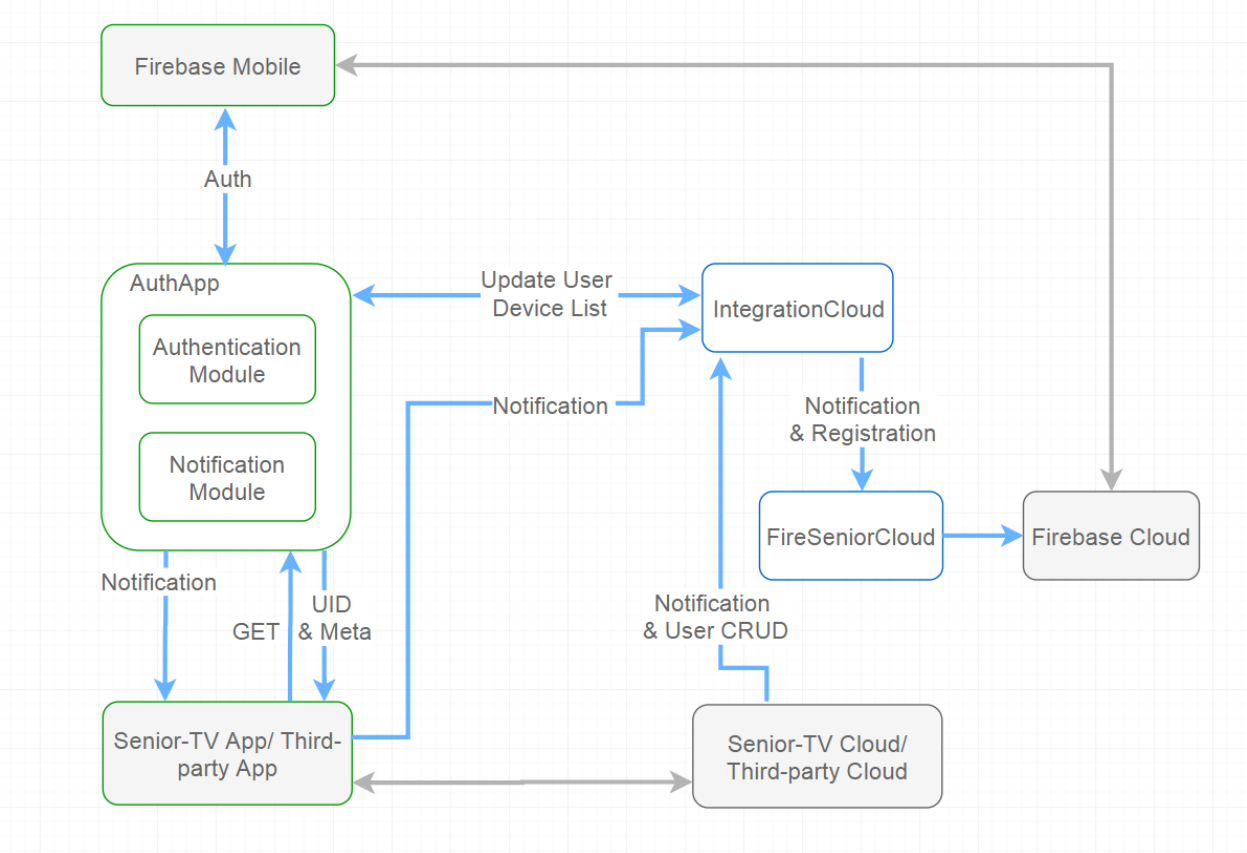


FIGURE 140. GENERAL ARCHITECTURE OVERVIEW

1. AuthApp: is an Android application responsible for user authentication and dispatching notifications to corresponding Android *ThirdPartyClients*.
2. IntegrationCloud: is the cloud service responsible for third-party User CRUD management, receiving notification messages from *ThirdPartyClients*, managing the device list for users and, getting user metadata.
3. FireSeniorCloud: this cloud service is responsible for receiving Notification Messages from *IntegrationCloud* and sending the notifications to the *AuthApp* and communicating with *Firebase* on user registration and deletion.
4. Firebase Mobile: is the client service for Google Firebase technology. It is part of Google Play Services upper 10.2.0 version.
5. Firebase Cloud: the Google Firebase cloud service.
6. UID: is the SENIOR-TV user identifier. It is used to identify the user across all the services (mobile or cloud).
7. DeviceToken: is a token which identify a specific user device.
8. UserDevice: is the Android Device where SENIOR-TV apps reside.

9. SENIOR-TV Cloud/Third-party Cloud: cloud services belonging to SENIOR-TV apps or third-party apps.
10. SENIOR-TV App/Third-party App: Android application running on the users' device.

8.1.1. IntegrationCloud Service

The scope of the IntegrationCloud is to store and manage the devices and user metadata belonging to certain UIDs and accepting third-party requests for user CRUD requests. The *AuthApp* will register devices and user metadata requests directly with the *IntegrationCloud*.

When a *Third-party app* needs to send a notification to a certain UID, the *IntegrationCloud* is firstly queried to retrieve the list of devices belonging to this specific UID. Then the device list and notification metadata are sent to *FireSeniorCloud*.

- **Inputs**: device Token and User UID from *AuthApp* for device list update, UID from *AuthApp* to return user metadata, user data from *Third-party apps* for users CRUD and, notification metadata from *Third-party apps* (everything at JSON format).
- **Outputs**: user metadata Response to *AuthApp*, notification metadata and user device list request to *FireSeniorCloud*, user data request to *FireSeniorCloud* for user register/delete at JSON format.

8.1.2. FireSeniorCloud Service

The scope of *FireSeniorCloud* is to receive Notification messages from *IntegrationCloud* and forward these messages to the corresponding user devices belonging to specific users and firebase user management.

- **Inputs**: notification messages and device list from *IntegrationCloud* and user data for Firebase (create/delete) at JSON format.
- **Outputs**: notification message to *FirebaseCloud*.

8.1.3. AuthApp Service

The *AuthApp* is an Android service responsible for authenticating the user on the SENIOR-TV platform and also managing the notifications. The users are registered and authenticated using an email and password.

The *AuthApp* is installed like an Android service in the Android TV devices. Therefore, after run it once it will automatically launch when the Android TV system start. In addition, the *AuthApp* will not have any interface screen, as in previous platform versions. In SENIOR-TV V3, the *AuthApp* just manages another TV apps requests. In order to log in and log out, to the SENIOR-TV platform users will use the *Client Ecosystem* app (see section 8.2. for more details).

Once a user is authenticated in a mobile or TV, the *IntegrationCloud* will receive a token which identify his/her device, and the user identifier (UID). Then the *IntegrationCloud* updates the device list with the token. Later the *IntegrationCloud* will send this user metadata based on saved UID.

The *Third-party apps* ask the user information to the *AuthApp* directly using Android Broadcast requests. Next section includes detailed information about how to integrate third-party apps with the *AuthApp*.

- **Inputs**:
  - Notifications from *Firebase*.
  - Authentication response from *Firebase* (with the user UID).
  - Broadcast intent from *Third-party apps* requesting UID and metadata.
- **Outputs**:
  - Notifications to corresponding Android *Third-party apps*.
  - UID and metadata to registered Android *Third-party apps*.

8.1.3.1. AuthApp Mobile

In order to use the SENIOR-TV mobile services such as the Tracker Mobile is necessary install the *AuthApp* app in the mobile devices. A particular version of *AuthApp* was developed for non-TV devices to log in without using *Client Ecosystem* (because the Ecosystem is just valid for TV devices). The *AuthApp* Mobile is installed like an Android app and it has two interface screens: one for start and another one for close the user's session (see images below).

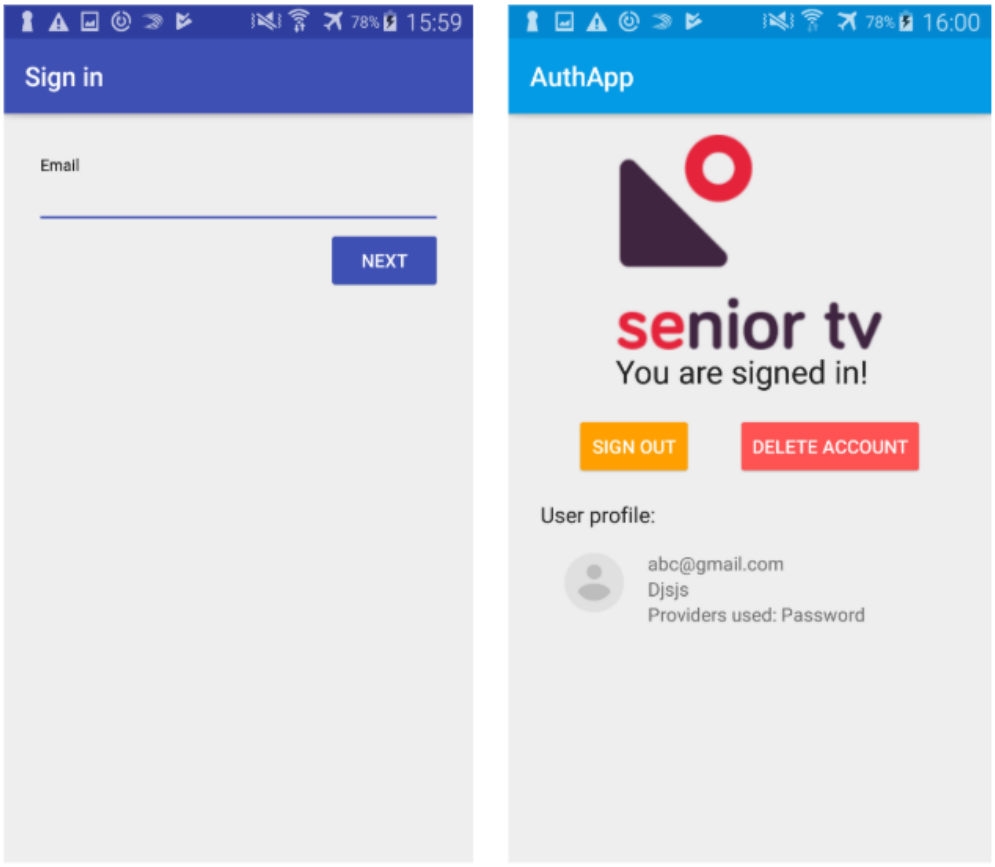


FIGURE 141. AUTHAPP SCREENS

8.1.3.2. AuthApp Third-Party Integration

Android supports applications developed by hybrid and native technologies (see section 2). This section describes how to integrate each kind of apps with the *AuthApp* service in order to access to manage users and notifications information.

1. Android Native apps

Third-party apps must implement *BroadcastReceivers* in order to be able to receive the notifications or the user UID on the Android device. Below the UID Broadcast and the Notification Broadcast receivers are included:

```
<receiver
    android:name=".UserTokenReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="SENIOR.TV.USER.UID"/>
    </intent-filter>
</receiver>
```

FIGURE 142. UID BROADCAST RECEIVER

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import org.json.JSONException;
import org.json.JSONObject;

public class UIDReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        String messageData = intent.getStringExtra("messageData");

        if(messageData == null || messageData.isEmpty()){
            Log.d("Token received: ","EMPTY message");
            return;
        }

        JSONObject messageDataJSON = null;
        try {
            messageDataJSON = new JSONObject(messageData);
        } catch (JSONException e) {
            e.printStackTrace();
        }

        if(messageDataJSON == null){
            Log.d("Token received: ","EMPTY messageData");
            return;
        }

        //convert specific JSON Message ( in this case token)
        JSONObject tokenMessageDataJSON = null;
        try {
            tokenMessageDataJSON = new
            JSONObject(messageDataJSON.getString("tokenMessageData"));
            Log.d("Token received: ", tokenMessageDataJSON.getString("token""uid")
                    + " " + tokenMessageDataJSON.getString("name")
                    + " " + tokenMessageDataJSON.getString("email")
                    + " " + tokenMessageDataJSON.getString("country")
                    + " " + tokenMessageDataJSON.getString("language")
                    + " " + tokenMessageDataJSON.getString("role"));
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

FIGURE 143. USER UID RECEIVER USING

```
<receiver
    android:name=".NotificationReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="SENIOR.TV.USER.NOTIFICATION"/>
    </intent-filter>
</receiver>
```

FIGURE 144. NOTIFICATION BROADCAST RECEIVER DECLARATION

```
import android.content.BroadcastReceiver;
import android.content.Context;
import  android.content.Intent;
import android.util.Log;
import org.json.JSONException;
import org.json.JSONObject;

public class NotificationReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        String messageData = intent.getStringExtra("messageData");
        if(messageData == null || messageData.isEmpty()){
            Log.d("Notification received: ","EMPTY message");
            return;
        }

        JSONObject messageDataJSON = null;
        try {
            messageDataJSON = new JSONObject(messageData);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        if(messageDataJSON == null){
            Log.d("Notification received: ","EMPTY messageData");
            return;
        }

        //convert specific JSON Message (in this case Notification)
        JSONObject notificationMessageDataJSON = null;
        try {
            notificationMessageDataJSON =
            new JSONObject(messageDataJSON.getString("notificationMessageData"));

            Log.d("Notification received: ",
                        notificationMessageDataJSON.getString("headerData")
                        + notificationMessageDataJSON.getString("bodyData"));
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

FIGURE 145. NOTIFICATION BROADCAST RECEIVER USE

The third-party client apps request the UID and metadata through a normal broadcast, and will receive the response in a listener

```
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import org.json.JSONException;
import org.json.JSONObject;

public class MainActivity extends AppCompatActivity {

   private static final String ORDERED_TOKEN_REQUEST = "ORDERED_TOKEN_REQUEST";

   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);

       Intent intent = new Intent();

      intent.setAction("ORDERED_TOKEN_REQUEST");
      intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
      sendBroadcast(intent);

   }
}
```

FIGURE 146. REQUEST UID TO AUTHAPP

2. Hybrid apps

Hybrid *Third-party apps* must import the custom GlobalBroadcasterPlugin in order to be able to receive the notifications or the UID on the Android device inside an Ionic Application. A GlobalBroadcasterPlugin using example is included below:

```
import { Component } from '@angular/core';
import { Platform } from 'ionic-angular';
import { StatusBar, Splashscreen } from 'ionic-native';
import { HomePage } from '../pages/home/home';
import { EventService } from "../services/event.service";
import { GlobalBroadcasterPlugin } from "GlobalBroadcasterPluginModule";
@Component({
  templateUrl: 'app.html',
  providers: [EventService]
})
export class MyApp {
  rootPage = HomePage;
  constructor(platform: Platform, private globalBroadcasterPlugin: GlobalBroadcasterPlugin) {
    platform.ready().then(() => {
      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
        StatusBar.styleDefault();
        Splashscreen.hide();

        globalBroadcasterPlugin.addEventListener("SENIOR.TV.USER.NOTIFICATION").subscribe(
            (event) => {
                alert("New Event Added\n\n" +
                    event.notificationMessageData.headerData + "\n\n" +
                    event.notificationMessageData.bodyData.substring(0, 50) + "...");

                console.log(event.notificationMessageData);
            },
            (err) => console.log(err)
        );
    });
  }
}
```

FIGURE 147. GLOBALBROADCASTERPLUGIN IN EVENTS SERVICE

Following an example of normal broadcast to request the UID and metadata and a listener to retrieve it. After installing the plugin in a hybrid project, the *GlobalBroadcasterPlugin* module can be imported as a component and the plugin methods are accessible.

```
import { Component } from '@angular/core';
import { Platform } from 'ionic-angular';
import { StatusBar, Splashscreen } from 'ionic-native';
import { Storage } from "@ionic/storage";

import { HomePage } from '../pages/home/home';
import { EventService } from "../services/event.service";

import { GlobalBroadcasterPlugin } from "GlobalBroadcasterPluginModule";

@Component({
  templateUrl: 'app.html',
  providers: [EventService]
})
export class MyApp {
  rootPage = HomePage;

  constructor(platform: Platform, private globalBroadcasterPlugin: GlobalBroadcasterPlugin, private storage: Storage) {
    platform.ready().then(() => {
      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
        StatusBar.styleDefault();
        Splashscreen.hide();

        globalBroadcasterPlugin.addEventListener("SENIOR.TV.USER.NOTIFICATION").subscribe(
            (event) => {
                alert(event.notificationMessageData.headerData + "\n\n" +
                    event.notificationMessageData.bodyData.substring(0, 50) + "...");

                console.log(event.notificationMessageData);
            },
            (err) => console.log(err)
        );

        globalBroadcasterPlugin.addEventListener("SENIOR.TV.USER.UID").subscribe(
            (event) => {
               // alert(event.tokenMessageData.name);
                console.log(event.tokenMessageData);
                storage.set("userEmail", event.tokenMessageData.email);
            },
            (err) => console.log(err)
        );

        globalBroadcasterPlugin.fireNativeEvent("ORDERED_TOKEN_REQUEST", {})
            .then(data => {
                console.log(data);
            });

        });
  }
}

```

FIGURE 148. RETRIEVE ON DEMAND THE USER UID

8.1.4. User, roles and relations API

The User API is the cloud System Integration service responsible to manage the SENIOR-TV user accounts and the relations between them. The user information is composed by the following fields:

- Name
- Email account
- Password
- Role: caregiver or senior.

- Language: English, Greek, Romanian, Slovenian or Spanish.
- Country: Default, Cyprus, Romania, Slovenia or Spain.
- Photo (optional).

The SENIOR-TV users are categorized in two types or roles: primary and secondary. Some SENIOR-TV services will have different available functionalities depending on the user role.

- **Primary** users correspond to seniors.
- **Secondary** are other user no senior: relatives, caregivers, care centers, etc.

Users will be able to related to another users. Two kind of relationships are included in the platform: friend and caregiver.

- **Friend**: is the default relation between two users and it means that one user knows to another one.
- **Caregiver** relation between a secondary user and a primary user.

The following list includes all the User API available functionalities, all of them are accessible via REST requests:

1. Register a user in the SENIOR-TV platform.
2. Read all the information of a specific user.
3. Get complete data of all the users.
4. Update user information.
5. Remove a user of the platform.
6. List the users of a specific country.
7. List the users with a particular role.
8. Create a relationship between two users (friend or caregiver).
9. Remove a relationship between two users.
10. Read the friend's information of a particular user.
11. Read the friend's information users cared by a specific user.
12. Get the caregiver list of a specific user.
13. Remove a user role.

8.2. Client TV Ecosystem

The Client Ecosystem is an Android app created in order to make easier the interaction with the SENIOR-TV platform from a TV device. Ecosystem main objective is showing the SENIOR-TV platform services organized by categories, hiding all particular characteristics of the Android TV device where it is installed.

The Ecosystem uses the *Integration services* to manage the users and service notifications data. The figure below shows the Ecosystem architecture:

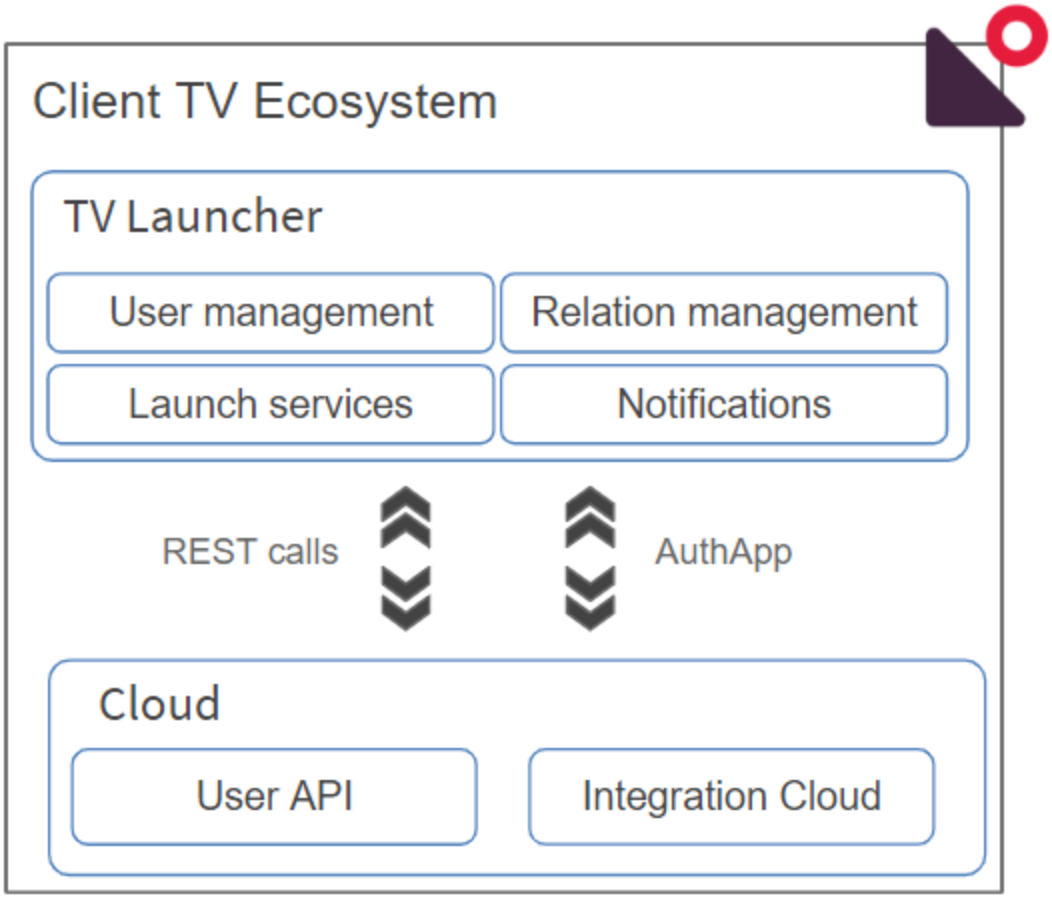


FIGURE 149. SENIOR-TV ECOSYSTEM ARCHITECTURE

These are the Client Ecosystem main functionalities:

1. User management: allows to register new users in the SENIOT-TV platform as well to initiate and close users' sessions.
2. Relationships management: to create and remove user friends.
3. TV services launcher: shows the SENIOR-TV services organized by categories, and allow to launch them from an interface adapted to seniors.
4. Notifications: shows the notifications sent by services to the registered users.

In the following subsections are included the functionalities details and the interaction with Integration services.

8.2.1 User management

In order to use the SENIOR-TV services (except games a Wikipedia) is necessary having a valid SENIOR-TV account. After starting up, the Ecosystem requests the device status to the AuthApp (figure 150):

- In case a user credentials are detected, the Category Menu screen is opened (figure 155).
- In other case, the Login screen is shown (figure 152).

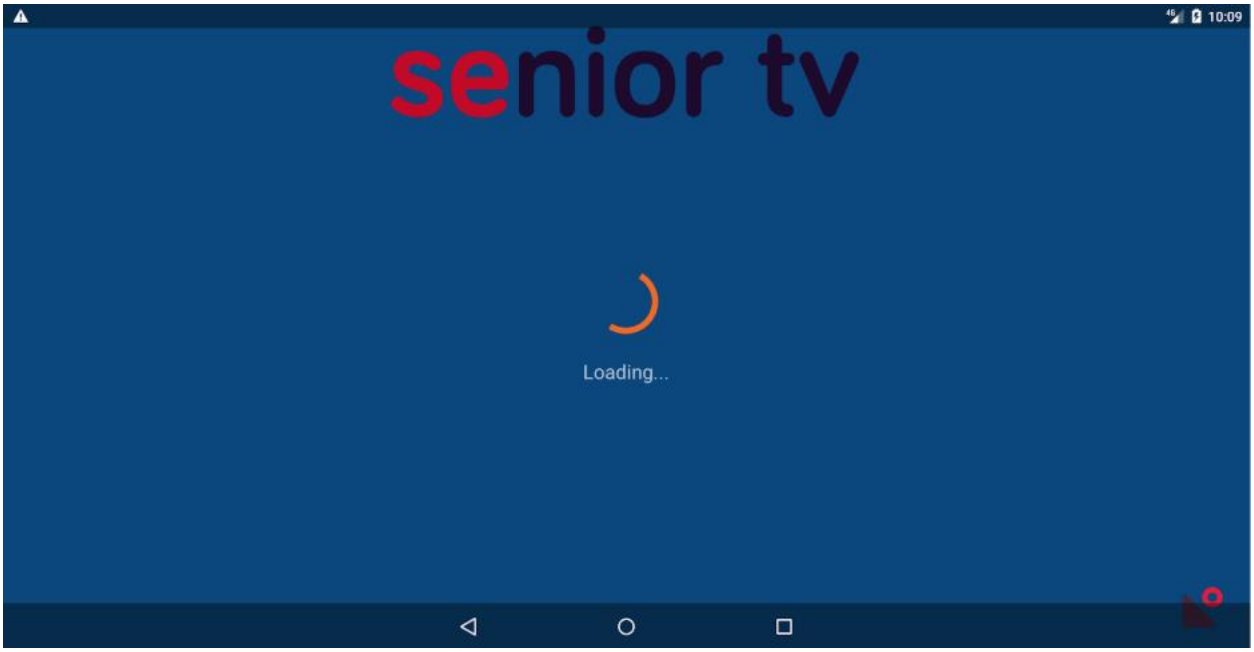


FIGURE 150. ECOSYSTEM - CHECKING USER ACCOUNT

From Login screen, users will be able to start a SENIOR-TV session using a mail and password previously registered in the platform. Ecosystem sends the user credentials to the *AuthApp* and it will communicate with the *IntegrationCloud* service to check if the credentials are valid. Later, the cloud response will send to the Ecosystem through the *AuthApp*.

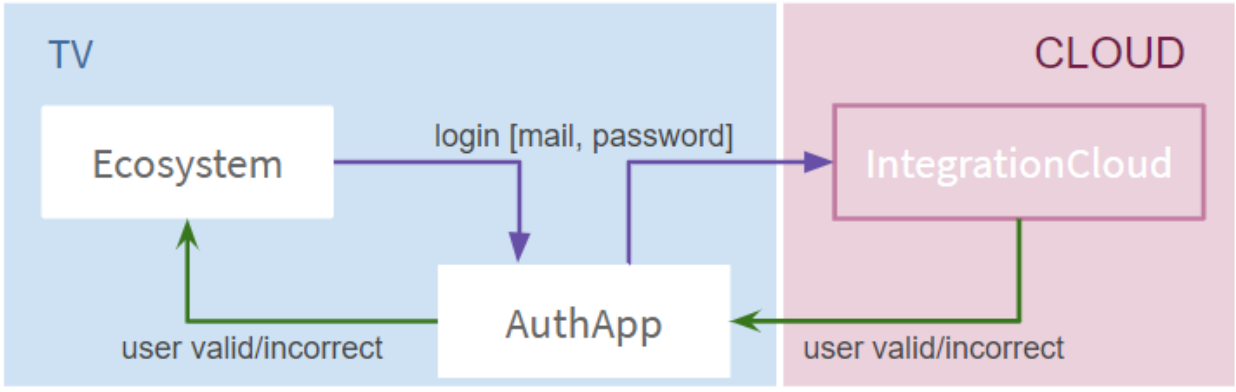


FIGURE 151. ECOSYSTEM - LOGIN PROCESS

When a user is logged in, his/her name will appear in the upper right corner of the screen. Selecting the user name field, the user session can be closed. The log out process generates a communication between Ecosystem and *IntegrationCloud* through the *AuthApp* similar than log in process.

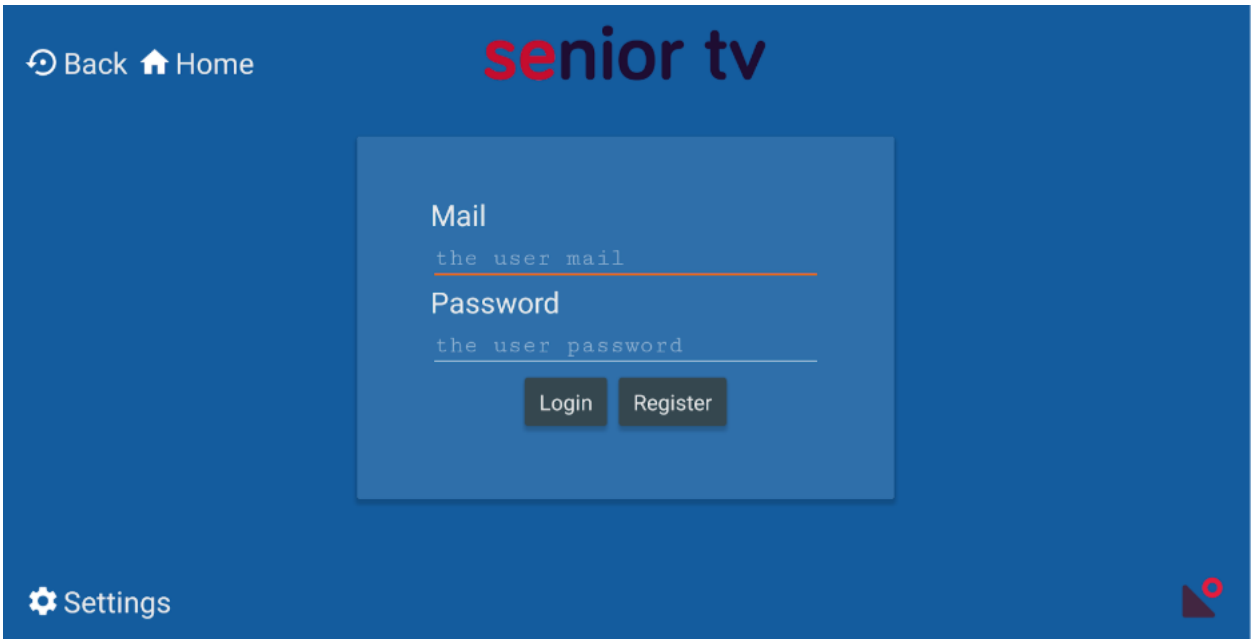


FIGURE 152. ECOSYSTEM – LOGIN

Users can create their own SENIOR-TV accounts from the Ecosystem application through the Registration functionality (figure 154). In order to create a new account, the Ecosystem sends a request to the *User API* with the new user information, as the next diagram shown:

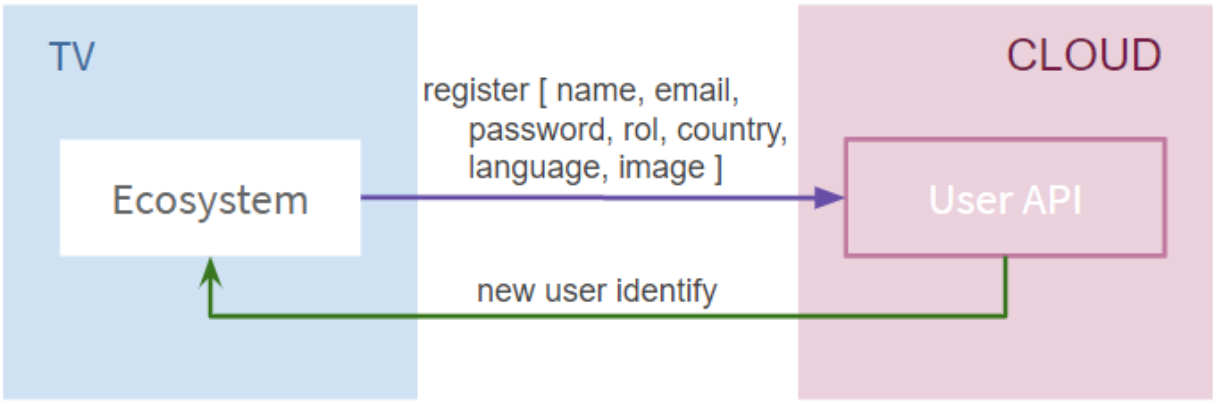


FIGURE 153. ECOSYSTEM - REGISTER PROCESS

FIGURE 154. ECOSYSTEM - REGISTER

8.2.2 TV services launcher

After staring a SENIOR-TV session, the platform services are displayed organized by categories in order to facilitate the search of a particular service. The SENIOR-TV categories are as follows:

- Information
- Games
- Social
- Entertainment
- Health

The following figure shows the category menu screen used by the seniors to navigate through the Ecosystem and use the different services and functionalities available.

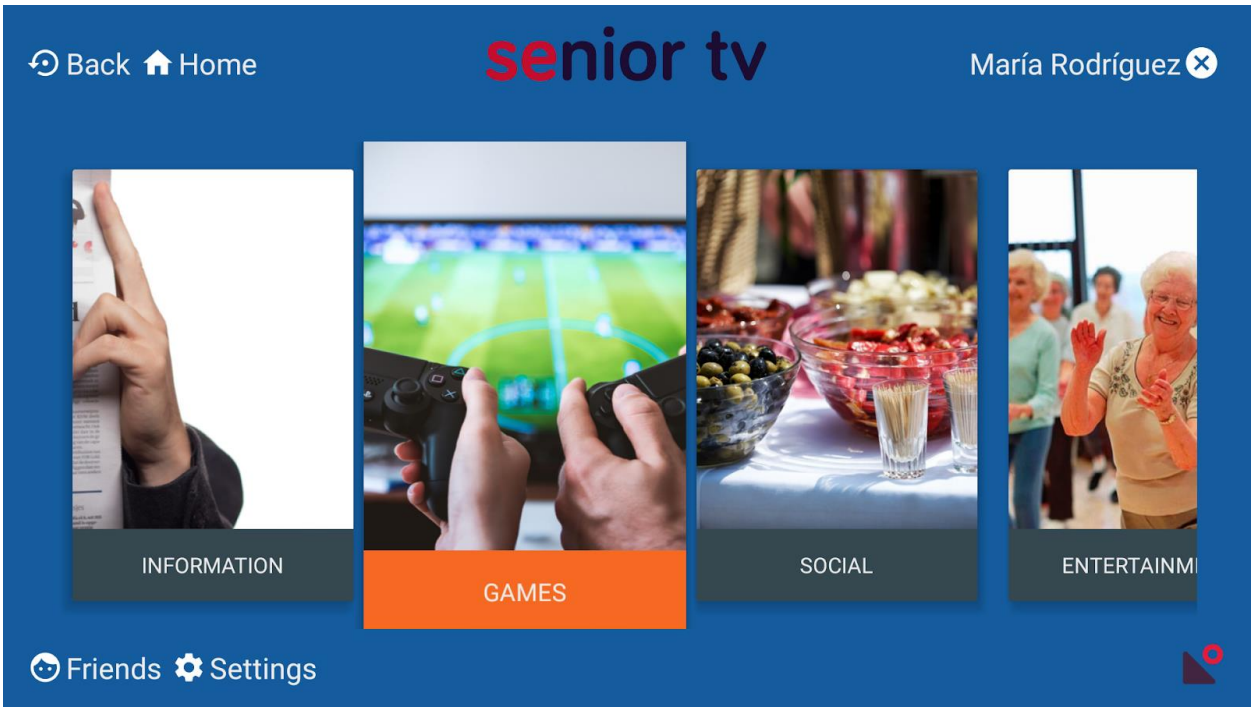


FIGURE 155. ECOSYSTEM - CATEGORY MENU

When a category is opened, the menu with the SENIOR-TV services belong to this category is shown. The following figure shows the "Social" category services: Video chat, Virtual Center and Social nets. Selecting an app card, the corresponded TV service app will start.

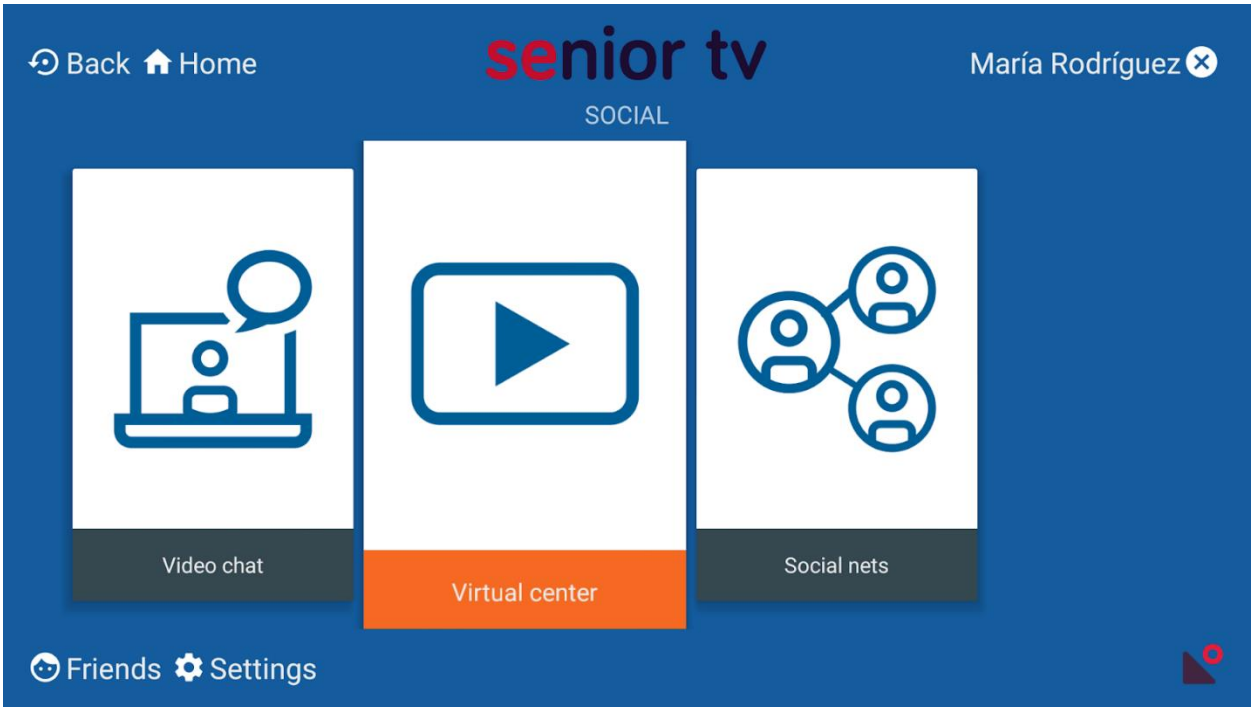


FIGURE 156. ECOSYSTEM - SOCIAL APPS MENU

8.2.3 Relationship management

The SENIOR-TV platform is a tool to reduce the senior's isolation keeping them in touch. The SENIOR-TV Ecosystem allows to create friend relationship between SENIOR-TV users. A logged user can manage their relationships using the Friends screen. The friends screen is divided in two sections (see next figure):

- On the left, the current logged user friends are shown.
- On the right, possible friends are listed. The friends are organized by the user country; therefore, the possible friends are all the SENIOR-TV registered users belong to the same country of logged in user.

In order to manage the users' relationships, the Ecosystem makes REST requests to the *User API*. The *User API* is the responsible to manage (create and remove) all the users' relationships.

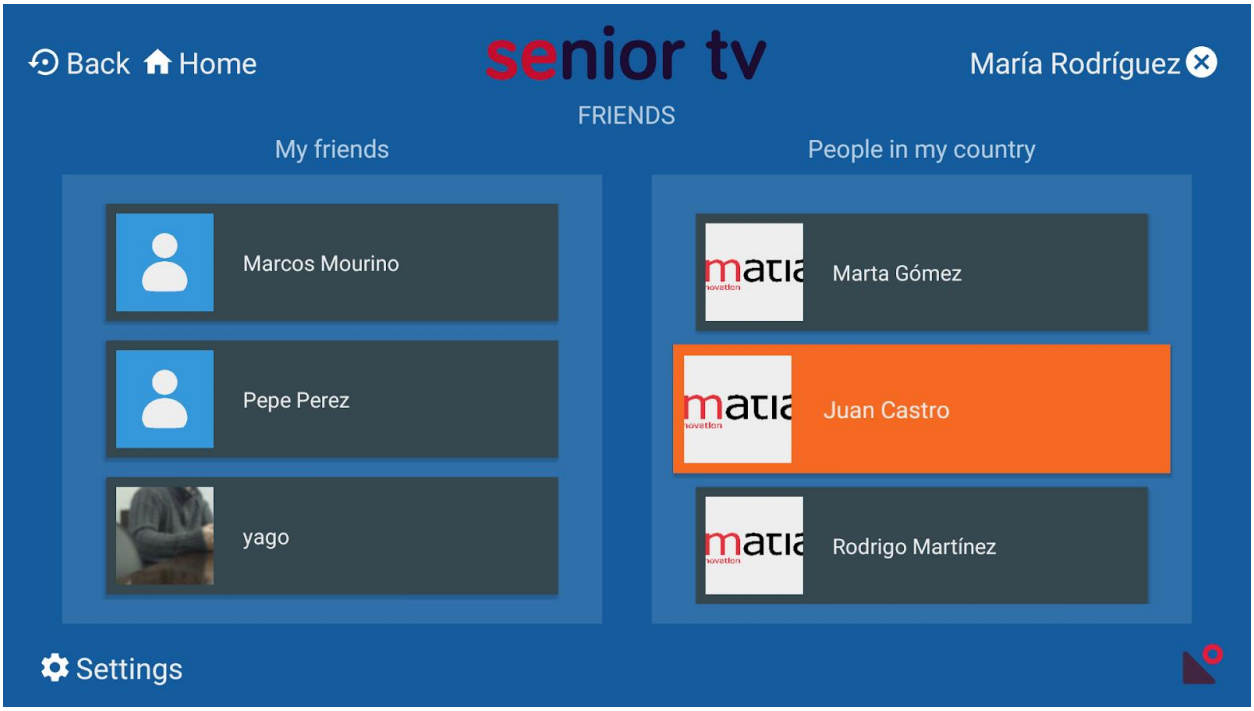


FIGURE 157. ECOSYSTEM - FRIENDS

8.2.4 Notifications

System integration has the capability to send notifications. This functionality can be used by SENIOR-TV services to notify users certain events o situations that would be able to interest for them. The notification process follows these steps:

1. *IntegrationCloud* module will receive the services notification and, it is the responsible to send this information to the TV devices. As indicated in section 8.1., every time user starts a session in a new device the *IntegrationCloud* register this event and update the user data.

2. Inside the TV devices, the *AuthApp* will receive all the notifications sent by *IntegrationCloud*.
3. Finally, the *AuthApp* asks to the *Client Ecosystem* in order to show the notification data as a message in the top of the TV screen (see figures 158 and 159).

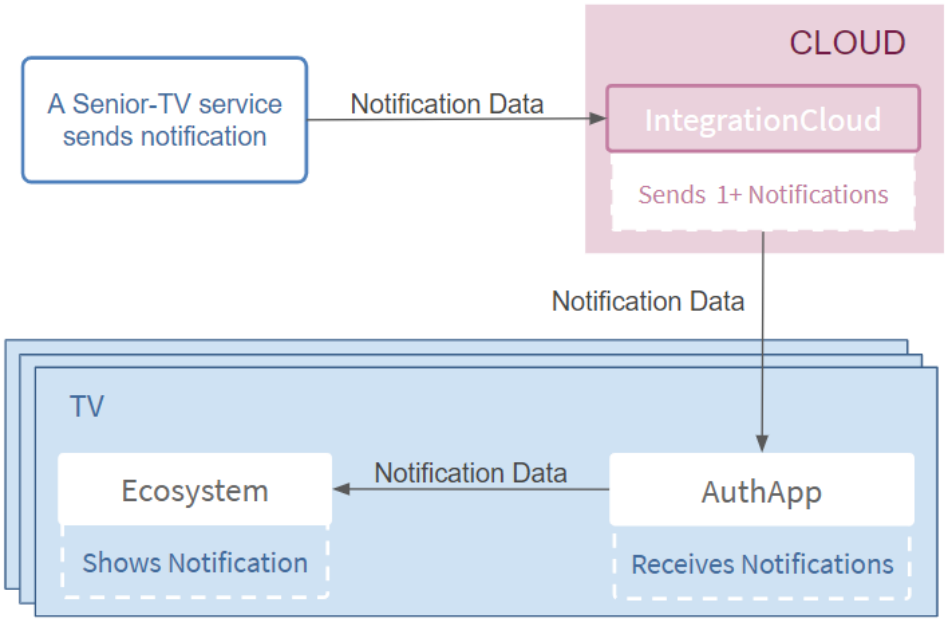


FIGURE 158. ECOSYSTEM - SEND A NOTIFICATION

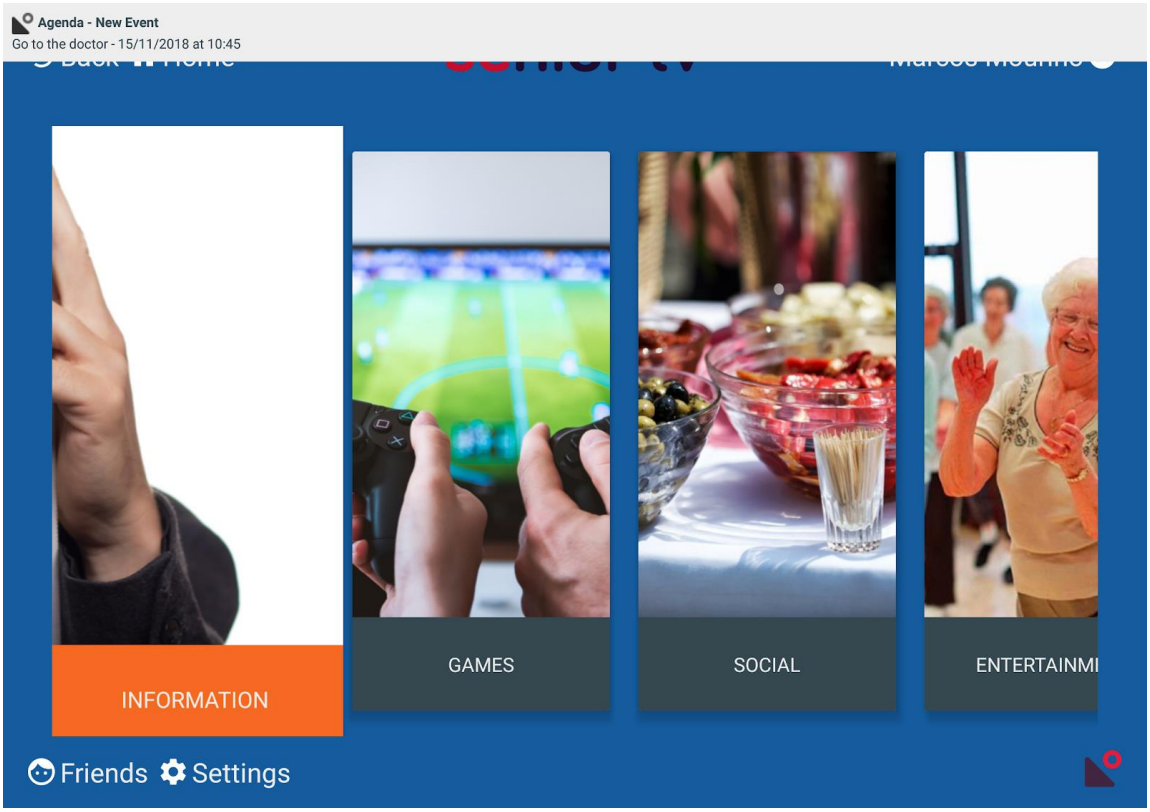


FIGURE 159. ECOSYSTEM - NOTIFICATION

9. Conclusions

As a result of the work made in WP2, the SENIOR-TV platform has been created. The decisions taken during the platform development have considered not only technological restrictions and limitations, but also, and especially, the elderly feedback. Seniors have been involved from the beginning in the non-technical decisions of the project. Their opinion has been essential for the selection of formal and informal services included in the platform, for the TV app interface design and, for the selection of the D-pad and touchable controls as the way to interact with SENIOR-TV.

From the very beginning, the SENIOR-TV platform base idea (using TV to bring formal and informal services of care and active aging to the homes of our elders) was well valuate by all stakeholders (seniors, relatives, caregivers and medical staff). Television is a well-known device for the elderly because they use the TV every day. This fact reduces the seniors' initial rejection of introduce new technologies in their home.

However, smart-TVs are not used in the same way as their traditional ones: seniors need a training time to know the new concepts and learn how to use the remote controls and new functionalities. It requires a seniors' effort and consequently increasing the frustration of the elderly and reducing the platform acceptance. In addition, the need to plugin a new device (STB) to their classical TV makes more difficult the access of the SENIOR-TV services, and reduces the acceptance, since the platform no longer looks like an extension of their classic TV and looks like another modern and complex device to use.

The final platform version includes services of diverse nature, as this document details. Those services fulfill the objective of improving the quality of life of the elderly and helping them to maintain a more active life both physically and mentally. For services selection, the seniors' feedback was considering through the WP1 workshops and the Pilot 1 and 2 results. However, some of the developed apps have resulted too complicated for their daily. Seniors reported comments like too many information in a screen or navigation difficulties access to some options, or useless functionalities. In order to bring these apps to market, simplest interfaces and a scope reduction are needed, focusing only in the most relevant functions identified by seniors.

Finally, a more significant involvement of the secondary stakeholders (relative, care centers, etc.) in the services' selection and functional design would have given a more attractive and open perspective to the platform for other market niches.

10. Annexes

10.1 SENIOR-TV Web Store

The SENIOR-TV Store is a web application to reduce the effort to install the SENIOR-TV platform in the devices used in pilots.

The first version of this web page was developed before second pilot and it was included all the SENIOR-TV V2 services (Agenda, Events, News, Tracker, Virtual Center and Weather) and the first version of Integration cloud system. The current version includes not only all the SENIOR-TV V3 services but also the complete Integration Ecosystem (client and cloud).

This web application has been developed in Angular. The contents are organized in cards. Every card contains a link to a SENIOR-TV resource. These are available the resource types:

- ".apk" files to install a SENIOR-TV service in an Android TV or mobile.
- Links to a SENIOR-TV web apps.
- User manuals.
- Trainer support videos.

The SENIOR-TV Store is divided in three sections which are accessed from the main menu in the top of the web page (see image below):

1. **Store App**: the SENIOR-TV ".apk" files for TV and mobile devices.
2. **Store Manual**: the TV and mobile support material (videos and user manuals)
3. **Web Service & Manual**: the links to SENIOR-TV web apps and their user manuals.

FIGURE 160. SENIOR-TV STORE TABS

1. Store App

This section contains all the ".apk" files to install the available services and the complementary Android services to obtain a complete installation of the SENIOR-TV platform (figure 161).

FIGURE 161. STORE APP SECTION

The page is divided in two subsections: TV and MOBILE. The applications can be installed independently. However, to ensure that all the platform necessary services are installed, just follow the order which the cards are displayed.

2. Store Manual

In the Store Manual section users can find all the support material to know how the TV and mobile services works. This support material is focused to trainers not to final user (seniors) and it is composed by a user manual and one or more videos. Each video explains how to use a particular function for each app.

Figure 162 shows the Agenda service section with a card to the user manual and four video cards (one for a specific Agenda TV functionality).

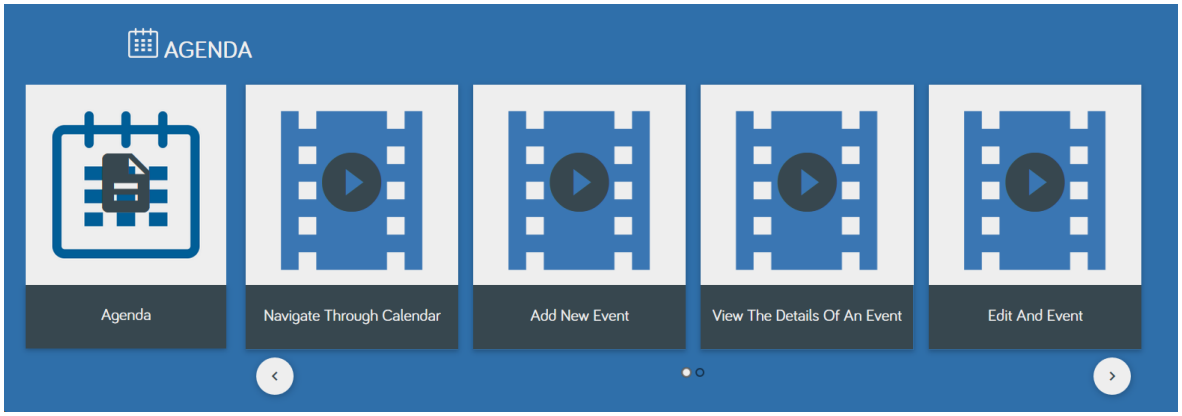


FIGURE 162. AGENDA MANUAL AND VIDEOS

3. Web Service & Manual

In the Web Service & Manual users will find the list of links to the complementary web applications of Agenda, Events, Video Channels, Virtual Center and Weather services (figure 163). In addition, the user manual of each web is included.

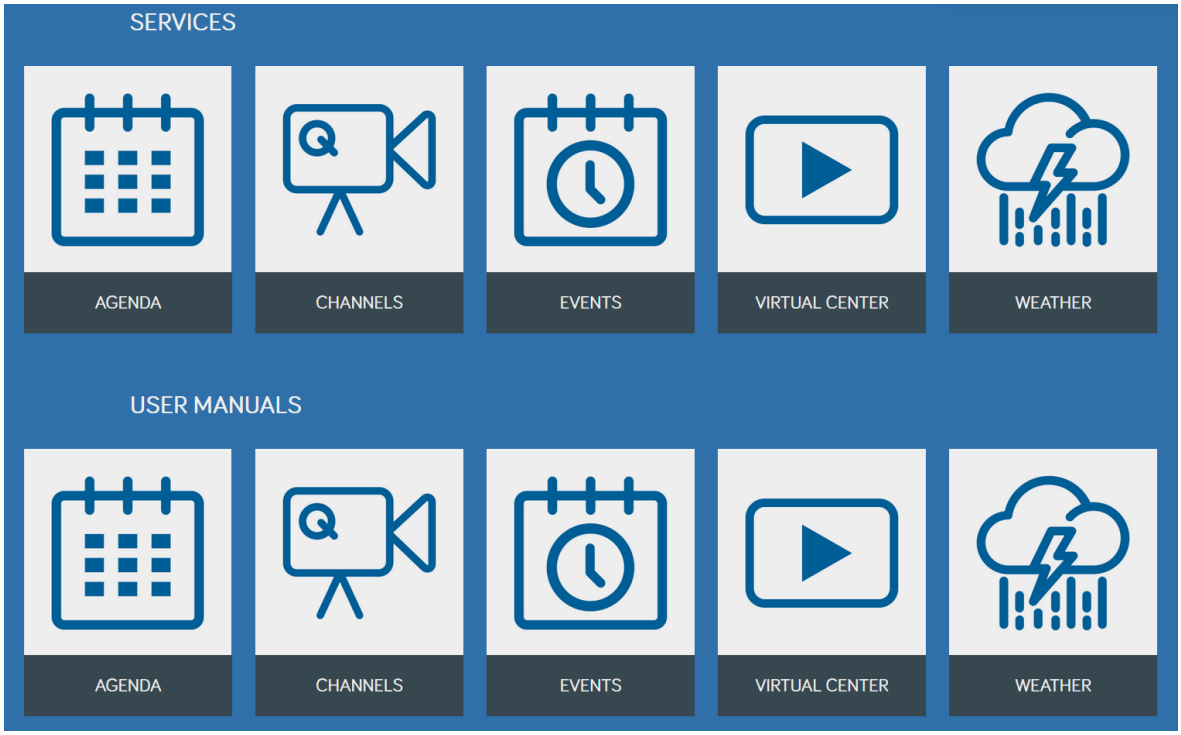


FIGURE 163. WEB SERVICES & MANUALS SECTION

aal-2014-171

SENIOR-TV

PROVIDING ICT-BASED FORMAL AND INFORMAL CARE AT HOME

Quality Checklist
Quality Control of D2.3

Peer Reviewer	
Reviewer	**Partner**
Luis Anido-Rifón	Advisory Board

CRITERIA	VERIFIED
1) Conformity to Standards and Project templates	
Logos (AAL, SENIOR-TV)	**X**
Project title, reference, author, version, revision, data	**X**
Mandatory statements (disclaimer)	**X**
Conformance to the standard structure required by EACEA (ex. Disclaimer, Executive summary, Acknowledgement, Introduction, page numbers, etc.)	**Disclaimer missing**
2) Language check (typing mistakes, grammar, etc.)	**X**
3) Coherence with objectives declared in the Technical Annex	
Obj. 1: To elaborate the project's Quality Plan following well-accepted methodologies tailored to the learning domain and based on a detailed description of projects objectives, success indicators and work plan.	**This document needs to be updated with the actual objectives related to this D2.3**
Obj. 2: To monitor all project activities and provide quality control of all project results as well as recommendations for improvements and identification of best practices.	**This document needs to be updated with the actual objectives**

	related to this D2.3
4) Reliability of data	
Information and sources well identified	**X**
Data and information are free from factual or logic errors	**X**
The analysis (if applicable) is reliable, i.e. previous studies have been sufficiently reviewed; qualitative information and quantitative data are balanced and appropriate	**X**
5) Credibility of findings	
Findings supported by evidence based on data analysis	**Not applicable**
Replicability of findings	**Not applicable**
6) Validity of conclusions	
Conclusions meet evaluation questions and information needs	**No**
Conclusions supported by proper evaluation findings	**No**
No conclusions missing according to the evidences presented	**No**

7) Please indicate any deviations from contractual conditions (WP objectives declared in the technical annex)

None

8) Comments/Suggestions for revision

The current version of the executive summary shows an enumeration with brief descriptions of the main conclusions. A more narrative approach should be taken to describe what should be expected to be found in this deliverable, what can be actually found and a short introduction to the main conclusions of the document. Current text should be conveniently re-written and transferred to the conclusions.

There is a very important overlapping with D2.2. The introduction should clearly identify what the new developments from year 3 are as well what updates have been made on previous outcomes. This should be accompanied by specific references where that information can be found in the document. If the vocation of this document is to substitute D2.2 this should be clearly stated.

The technical quality of the deliverable is good and shows an extensive work done in WP2. However, it is not clear how other WPs have influenced the strategy and decisions made in WP2. After two pilots, one may expect to find an analysis of feedback from end-users and other experts in the project and arguments supporting the work presented in this document beyond addressing what has been promised in the Description of Work.

9) Implementation of revisions/modifications suggested and explanation for eventual rejections (performed by the Responsible of the Deliverable)

Feedback from WP3 (from second cycle) and WP1 (from third cycle) needs to be included.

Conclusions requires to include, in addition to the main conclusions, recommendations for potential future exploitation of the project results.

10) Deliverable accepted

☐ YES

X NO

If NO, please state reasons: See recommendations above

aal-2014-171

SENIOR-TV

PROVIDING ICT-BASED FORMAL AND INFORMAL CARE AT HOME

Quality Checklist
Quality Control of D2.3

Peer Reviewer	
Reviewer	**Partner**
Cibrán Ledo Peláez	IMATIA

CRITERIA	VERIFIED
1) Conformity to Standards and Project templates	
Logos (AAL, SENIOR-TV)	**yes**
Project title, reference, author, version, revision, data	**yes**
Mandatory statements (disclaimer)	**yes**
Conformance to the standard structure required by EACEA (ex. Disclaimer, Executive summary, Acknowledgement, Introduction, page numbers, etc.)	**yes**
2) Language check (typing mistakes, grammar, etc.)	**yes**
3) Coherence with objectives declared in the Technical Annex	
Obj. 1: To elaborate the project's Quality Plan following well-accepted methodologies tailored to the learning domain and based on a detailed description of projects objectives, success indicators and work plan.	**yes**
Obj. 2: To monitor all project activities and provide quality control of all project results as well as recommendations for improvements and identification of best practices.	**yes**
4) Reliability of data	
Information and sources well identified	**yes**
Data and information are free from factual or logic errors	**yes**

The analysis (if applicable) is reliable, i.e. previous studies have been sufficiently reviewed; qualitative information and quantitative data are balanced and appropriate	**yes**
5) Credibility of findings	
Findings supported by evidence based on data analysis	**yes**
Replicability of findings	**yes**
6) Validity of conclusions	
Conclusions meet evaluation questions and information needs	**yes**
Conclusions supported by proper evaluation findings	**yes**
No conclusions missing according to the evidences presented	**yes**
7) Please indicate any deviations from contractual conditions (WP objectives declared in the technical annex)	
No deviation from contractual conditions already notified to CMU regarding WP2	
8) Comments/Suggestions for revision • The Conclusions section must be more elaborate. It must include a brief information regarding the technology usability and acceptance.	
9) Implementation of revisions/modifications suggested and explanation for eventual rejections (performed by the Responsible of the Deliverable)	
10) Deliverable accepted **X YES** ☐ NO	
If NO, please state reasons:	