FOSTERING SOCIAL INTERACTION OF HOME-BOUND AND LESS EDUCATED ELDERLY PEOPLE

Complete Service Implementation and Documentation

Deliverable No.		**D2.3**	
Work Package No.	**WP2**	Workpackage Title	**Social-Interactions Components Prototype**
Authors (per company, if more than one company provide it together)		**CNR, FUB, UNIMIB, FIMI**	
Status (F: final; D: draft; RD: revised draft):		**F**	
File Name:		**EASYREACH Deliverable D2_3**	
Project start date and duration		**01 November 2010, 28 Months**	

List of abbreviations

DOW	Description Of Work
ER	Easy Reach
STB	Set Top Box
RCS	Remote Control Server
ERC	EasyReach Client
SIF	Social Interaction Front-end
SIB	Social Interaction Back-end
WSCM	Web Service Communication Manager
PA	Personal Assistant
MC	Model Controller
WS	Web Services
SOA	Service Oriented Architecture
RPC	Remote Procedure Call
WSDL	Web Service Description Language
UDDI	Universal Description Discovery and Integration
SOAP	Simple Object Access Protocol
EJB	Enterprise Java Beans
EIS	Enterprise Information System

Table of contents

EXECUTIVE SUMMARY

The present deliverable has the objective of describing the final design of the EasyReach system, motivating both the choices behind the technology solutions adopted in the development process, and the reasons behind the introduction of modifications of such solutions with respect to the project's initial technical design. In particular at start the document a short history of main technical decisions making esplicit their rationale.

This document provides a general view of both the low-level and high-level services that the current implementation of the EasyReach system offers to the users, and presents an overall description of the architectural elements of the EasyReach system, according to the following schema:

- presentation of the overall system architecture, and description of all the architectural software and hardware components as well as of the interconnections among them.
- description of the web services technology used to implement the network communication services, together with the motivations that guided such choice.
- description of the EasyReach services in a bottom-up fashion, starting from the presentation of the low-level services that have been realized according to the project's original objectives of simplicity of use for a pre-digital divide user basin.
- presentation of the basic services described at the previous point, as a set of building blocks that are finally exploited to compose the high-level social services. Such social services are described by providing some examples of typical utilization on behalf of an elderly user, enriched with a visual walkthrough that guides the reader through a selected series of steps that best represent the typical exploitations of the system's most common functionalities.
- finally, the document ends with the presentation of some concluding remarks concerning the feedback obtained from the latest public demonstration of the system, and the appreciation of the EasyReach idea both from the scientific community and the stakeholders that have been exposed to system during the official demonstration at the AAL Forum 2013.

1. Introduction: a brief summary of the evolution of the EasyReach Project

The main objective of the EasyReach system is encouraging socialization among elderly adults who, for various reasons, may be forced at home for prolonged periods of time. The basic idea that has been pursued to reach this goal is the design of an ad-hoc social network that was particularly tailored on the elderly users' requirements and preferences. One of the main difficulty encountered during the development of the proposed technical solutions stemmed directly from the biggest ambition of the EasyReach initiative, i.e., to provide easy access to ICT-based technological solutions to a *potentially pre-digital divide* basin of users. This extremely important requirement provided the basic guidance from the very beginning of the project, and acted as the main inspiration for the selection of the system hardware, as well as for its interaction with the software solutions, leading to the choice of utilizing commonly well-known appliances (e.g., an ordinary TV set), non-intrusive equipment (e.g., a Set Top Box - STB) and/or synergetic system-user interaction methodologies (e.g., a gesture-driven graphical user interface controlled through a familiar device such as a remote control).

The user-centered design approach that has been followed throughout the project has often provided significant guidance for all the issues related to system's usability and acceptability, which justified some of the modifications that have been applied during the development of the project. Also, some of the applied changes have been motivated by the results of the official mid-term project review meeting that took place in Brussels on 08/03/2012. After such meeting, the EasyReach System has in fact been refactored according to the feedback received from the reviewers in order to get it closer to the EasyReach DOW and also to facilitate its convergence towards a stable release within the limited amount of remaining available time.

Figure 1 presents a sketch of the original EasyReach System's architecture as it was initially devised. Comparing this architecture with the new one depicted in Figure 3, an increase in architectural modularity can be clearly appreciated. Among the most meaningful modifications to the original design, the software architecture internal to the SetTopBox has been slightly modified to the aim of clearly separating the "Client part" from the "Server part", where all the user's information are stored.

Also, as will be made clearer in the following sections, a key decision has been the one of interrupting the development based on the Elgg social network engine, in favor of a more specific and effective social engine, expressly built for the EasyReach project. In fact, the Elgg tool turned out to require too many refinements in order to serve the EasyReach purposes. The current solution, redesigned from scratch, is much lighter and flexible enough for the project objectives.

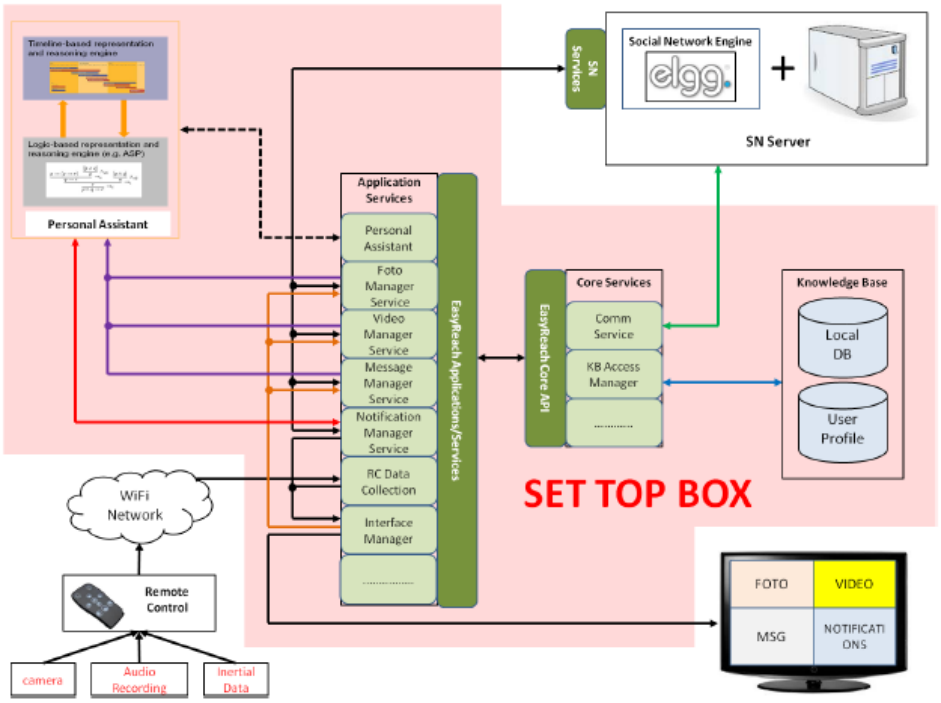

Figure 1: The original EasyReach System's Architecture

In the following, we present a brief summary of the EasyReach System's evolution in time, by highlighting the milestone dates associated with the most meaningful development updates and/or modifications with respect to the original design.

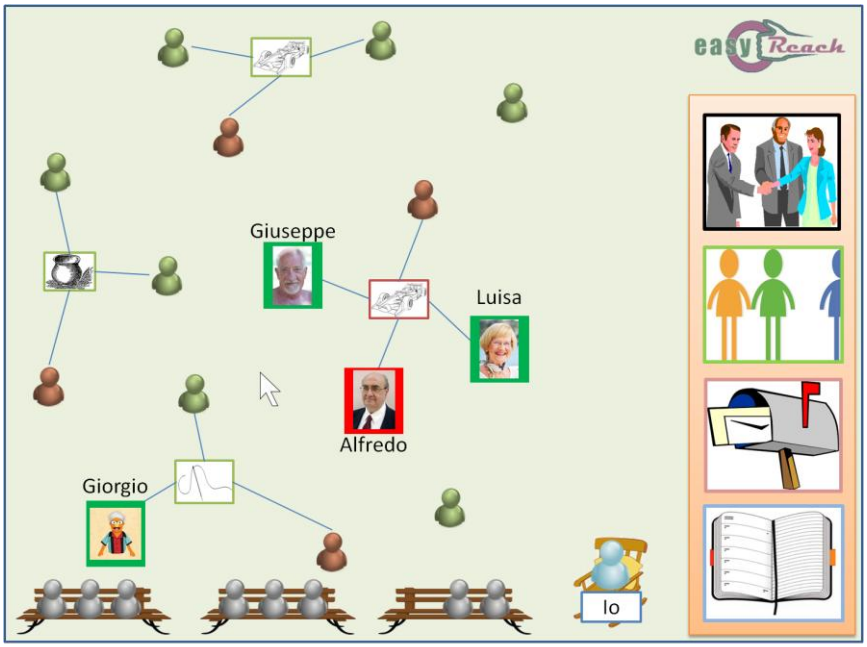

Figure 2: The original EasyReach "Square Metaphor"

- Immediately after the mid-term review meeting, the problem of the general interaction of the old person with the "social channel" has been addressed in a restricted meeting of people working in WP1 and WP2 that took place in Rome on 18/04/2012. At that time, we were still clearly missing a "system interaction solution" that could be of general use. As a consequence, the "Square metaphor" was analyzed and introduced in the System. Indeed, the squares of most small villages around Europe represent the aggregation points where the people gather in order to socialize. Since encouraging socialization is the main objective of the system, the same idea was chosen for the main screen of the social interaction within the EasyReach environment: a virtual place where to meet people and/or groups of people, i.e., where users can interact with other users, e.g., talk about interesting topics and share their experiences, like in a real square. Figure 2 presents the initial idea of the square metaphore as it was originally presented.

- During the general meeting held in Rome on the 22-23/05/2012, the "Square metaphor" has been presented and explained to the partners and to some of the potential users of the System. Also, a first review of the System's architecture and User Interface has been presented in order to receive feedback before its detailed implementation would start.

- In the 2-3 months following the previous meeting held in Rome, the work has focused on the implementation and integration of the Front-end System Environment (User Interface – TV – Remote Controller) and the Back-end System Environment (Personal Assistant module – Server and Web Services for the communication).

- In the following July, a series of meetings with the users have taken place in order to receive some feedback for the developers. At the beginning of August, the Square Metaphor has been slightly re-focused in favor of a simpler Group-oriented environment: the modification was decided to simplify the System and the Front-End side, in order to give the user a less complex System to interact with, yet with all the functionalities and the services of the original one. In fact, the initial Square instantiation was generating a too much complicated Front-End Environment System; conversely, the newly proposed system would allow the user to take advantage of the same socialization services of the original system through a much simpler interaction environment that could be further simplified after a long term use test. The User Interface was revised accordingly for the same purpose; the new version of the User Interface will be thoroughly described in Section 4 of the present document.

- A demo of the System was presented at the AAL Forum at Eindhoven (24-25-26-27 September, 2012). A first initial integration with a prototype of the remote controller

was realized for the demo, to the aim of demonstrating the closed utilization loop, despite the system being not completely implemented.

- In November 2012, all the modules presented at the AAL Forum were fully integrated, resulting in the final version of the project capable of implementing the high-level social services – ultimate objective of the EasyReach Project - which will be described in Section 5.

After the system integration achieved in November, an intensive set of evaluation steps have been performed in the form of two pilots, in order to assess the developed technological solutions in real life environments. The relevant characteristic of these pilots has been to conduct field trials involving real users, in order to (i) evaluate system usability by the elderly people in different environments (e.g., home, senior center); (ii) evaluate user experience and user acceptance of the ICT solution developed in the project; (iii) evaluate the effectiveness of the EasyReach solution in terms of social inclusion and improvement of quality of life (see [1,2] for a more detailed description of the Pilots' results). In the rest of the document, the ultimate version of the EasyReach system will be described, and some of the previously mentioned modifications with respect to the original design will be further commented.

2. The Updated EasyReach Service Architecture

Figure 3 presents the definitive general architecture of the EasyReach system. The figure is composed so as to provide a clear idea of all the main components as well as the interactions among them. As the figure shows, the main element is represented by the Set Top Box (STB), which provides the "gluing factor" among all the hardware components. The current organization allows the user to access the system functionalities and join the EasyReach network by enabling and simplifying the user's interaction with a TV set and the Internet. As Figure 2 shows, within the EasyReach organization, the STB is mainly used for (i) reasoning on the data gathered from the Remote Control (RC) in order to correctly interpret both user input and gestures, (ii) processing the multimedia contents recorded by the user, (iii) directly controlling the TV set, (iv) running the core social components, and (v) managing the Internet connections and contents. More specifically, the STB can be decomposed in two main modules, the Remote Control Server and the EasyReach Client, which will be described in the following sections.

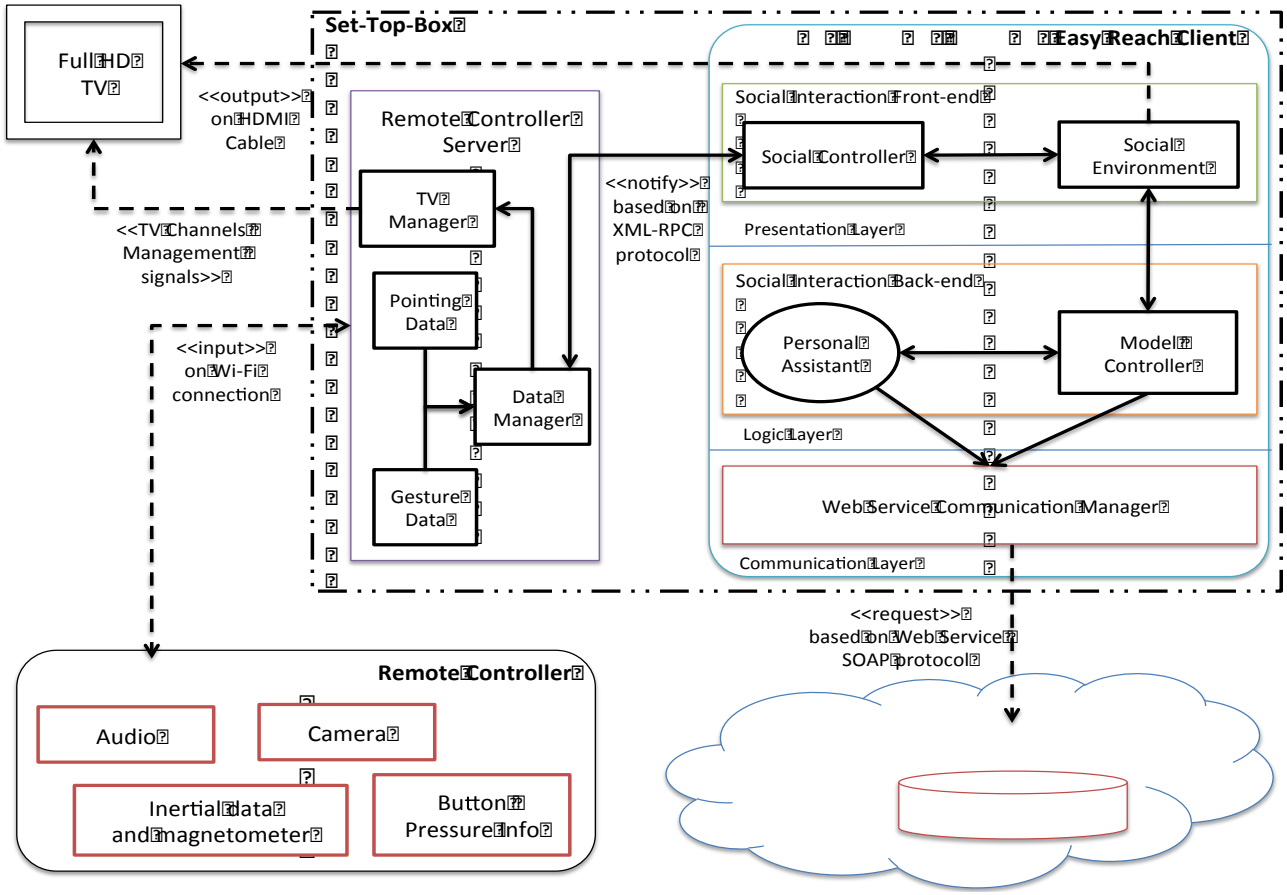

Figure 3: The definitive System Architecture

2.1. The Remote Control Server module

The Remote Control Server (RCS) contains all the software modules and drivers necessary to manage the interaction with the RC, in terms of signals and data exchange. In particular, the data flow coming from the RC is received by the RCS and interpreted according to the data type (*Pointing Data* or *Gesture Data*), as follows. The inertial data coming from the RC that are to be used to drive the GUI pointer (Pointing Data), will be reasoned upon differently by the *Data Manager* submodule from the inertial data that are to be used to recognize the user's gestures (Gesture Data). The Data Manager is also in charge of acquiring the multimedia data flow (not shown in the picture), i.e., all the pictures, audio and/or video contents recorded by the user. Moreover, the TV Manager submodule is in charge of re-directing to the TV set all the ordinary channel management signals coming from the RC. In this way, the RC can be used exactly as any familiar remote control to change channels, to set the volume, etc, every time the user is watching the regular TV channels (i.e. she/he is outside the dedicated EasyReach channel).

2.2. The EasyReach Client module

The EasyReach Client (ERC) is the STB module which is in charge of creating and managing the personal environment within which all the EasyReach services are exploited to allow the user's activities within the system. The ERC is basically follows a stratified design, being basically divided in three intercommunicating layers, i.e., the *Presentation* layer, the *Logic* layer, and the *Communication* layer. Each layer is associated to a different software subcomponent, i.e., the Social Interaction Front-end, the Social Interaction Back-end, and the Web Service Communication Manager, respectively.

1. The Social Interaction Front-end (SIF) is in charge of offering the user a suitable Social Environment as well as all the means to easily access it and interact with it, by means of the EasyReach Graphical User Interface, a graphical front end particularly devised to meet the needs of a potentially pre-digital divide user basin. As Figure 3 shows, the *Social Controller* subcomponent of the SIF continuously exchanges data with the RCS Data Manager previously described, thus allowing a direct control of the Easyreach GUI by the sole means of Remote Control input. The services offered by the SIF can be enjoyed directly on the TV set, as the STB is connected with the TV through a HDMI cable. In other words, the EasyReach GUI will be directly projected on the TV, thus allowing the user to perceive the EasyReach channel exactly as any other ordinary TV channel.
2. The Social Interaction Back-end (SIB) is the subcomponent which manages the information used to drive the Social Environment (i.e., the EasyReach GUI), therefore representing the core reasoning engine of each EasyReach Client. It is basically composed of two modules, the Model Controller and the Personal Assistant (a detailed description of the Personal Assistant module can be found in [3]). The Model Controller (MC) is in charge of acquiring and reasoning on all the data that are significant for the creation and management of the user model; the results of this reasoning process will directly reflect on the Social Environment. The user model is composed of all the information that represent and identify the user; all the information related to the EasyReach users can be contained in one or more dedicated DataBases that compose the EasyReach Cloud. The MC can be basically seen as the software module that is in charge of managing the DB views that are associated to each user.
3. The Web Service Communication Manager (WSC) is the subcomponent that is responsible for providing high-level communication facilities by means of which the client can interact with server-side elements (the EasyReach Cloud). The EasyReach Cloud realizes the EasyReach user network and its main role is to abstract the data localization, thus masking the possible data fragmentation and ensuring a proper EasyReach services availability.

2.3. Integration of enhanced functionalities in the EasyReach Remote Control

This section is dedicated to provide a better insight of the RC hardware component, whose basic inner components are shown in Figure 4. Instead, Figure 4 provides a realistic representation of the remote controller device, giving a true idea of its dimensions and showing its enclosure as well as its internal electronics. The main characteristic of the remote controller is that of it gathers inertial and multimedia data to capture the behavior of the user. The remote control includes a complete three-dimensional inertial unit (with accelerometers, gyroscopes and magnetometers), a camera, a microphone, a keyboard and a rechargeable battery.

Figure 4: The EasyReach Remote Control's internal electronics

The inertial data is used to track user movements or recognize particular gestures performed by the user for interacting with the system. As explained in Section 2, a software component, running in the STB, converts the low-level inertial data to pointer and gesture data in order to recognize higher level features such as particular movements (e.g. moving the remote left, right, up and down) and rotations (e.g. clockwise and counter-clockwise). The camera and the microphone are used for acquiring multimedia contents. In order to interact with others and share data, the remote control can be used by the user either for taking photos or for recording audio and video messages. Moreover, the remote control includes a simplified keyboard that can be used for performing simple actions such as turning the system on/off and for controlling the basic TV functions.

During the EasyReach system's development, several prototypes of the remote controller have been realized, starting with a software version based on a smartphone. Additionally, several hardware devices working during time on cost reduction and energy optimization have been developed. The current device is low-cost and energy efficient. The tests that have been performed on the device show that its battery can last for a few days of continuous usage. For more detailed information about the RC device, the reader can refer to [4].

3. Behind the EasyReach System activities

All the EasyReach activities have been built on top of the functionalities exposed by the EasyReach Cloud element of the architecture (see Figure 3). The EasyReach cloud represents the server-side component responsible for providing access to System information (DB) and communication capabilities among all the EasyReach clients.

In particular, the EasyReach Cloud exposes a set of facilities in the shape of Web Services. The clients exploit these services by exchanging a set of messages built on top of well-known standards.

The reasons that led us to choose Web Services technology as the main mechanism for exchanging information between clients and server is the great flexibility and interoperability this technology supports. Indeed, the Web Services technology allows us to define a set of self-contained elements with clearly defined interfaces and responsibilities. The EasyReach system functionalities are built as complex business processes obtained by different composition of the realized services. Another important feature of this technology is that it is based on precise standards, thus allowing us to "open" the EasyReach system functionalities to different platforms, and therefore facilitating their cooperation.

For all the previous reasons, it has been decided to abandon the initially selected idea of using Elgg as a social networking engine to implement the system's communication infrastructure in favor of the Java EE + Web Services. Despite the high versatility of the Elgg platform, after receiving the first feedback from the users during the first stages of the implementation, we came to the conclusion that the best solution to obtain the EasyReach objectives would require a less "PC-based" approach. In fact, all the interactions in Elgg are naturally textual, i.e., based on keyboard input, while the first interviews with potential users have soon identified the keyboard as a strongly "feared" device. In other words, while Elgg is a rather complete tool for implementing "ordinary" social networks, its strong PC-based characteristics was making it less suitable to realize services specifically tailored to be exploited with little or no keyboard utilization.

Before describing the EasyReach Cloud Web Services in detail, we provide a brief introduction to Web Service technology in order to explain its characteristics and the reasons that led us to choose this technology.

3.1. The current Web Services technology

The service-oriented middleware is the emerging technology for developing and integrating distributed applications, and Web Services (WS) is the main technology in this context. The service-based middleware is an evolution of distributed architectures. It aims at supporting interoperability among heterogeneous components founding communication on open and well-known standards. In this way it is possible to reuse already existing services for different purposes in order to realize complex business processes (also reducing development costs).

A Web Service in a Service Oriented Architecture (SOA) is a mean for exposing a cohesive set of business functionalities (application logic) in order to make them accessible through standard web technologies; therefore, SOA clients can access services over the network by means of standard messages realizing a generalized RPC call mechanism.

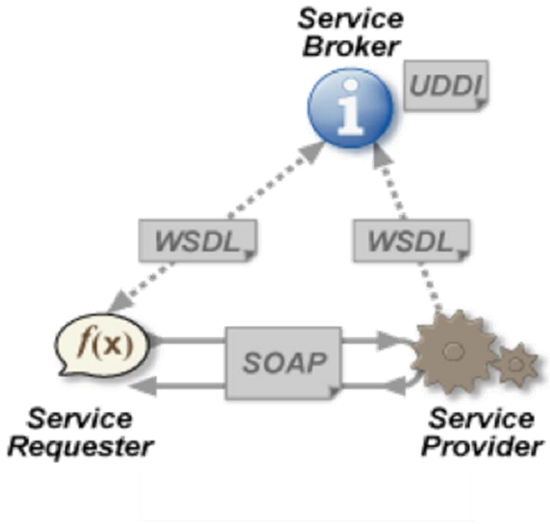

Figure 5: The Web Service middleware

More precisely, the W3C defines a Web Service as:

> *"[...] a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."*

Web Services middleware is built upon a set of XML standards that allow to expose, search and access services over the network. With respect to Figure 5, it is possible to identify the following main standards for WS:

- Web Service Description Language (WSDL) is a language for defining the interface of a Web Service;
- Universal Description Discovery and Integration (UDDI) is a standard that allows to search available Web Services;
- Simple Object Access Protocol (SOAP) allows the interaction among services and client defining the organization of documents in order to exchange structured information and data.

3.2. The EasyReach Cloud: A Web Service container

The EasyReach Cloud is responsible for providing access to system information by means of a set of services that composed together allow clients to build the EasyReach "Social Services".

The EasyReach Cloud has been organized according to the general purpose Java Enterprise Edition (Java EE) layered architecture depicted in Figure 6. With respect to this figure, the EasyReach Cloud can be seen as a Java EE Server where, the Business Tier is composed by several Enterprise Java Beans (EJBs), which realize the business logic by locally providing access to system information encapsulated into the Enterprise Information System (EIS) Tier. Moreover, the Web Tier in Figure 6 has been replaced by a Service Tier composed by a set of Web Services (WSs) exposing system functionalities externally to the Client Tier.

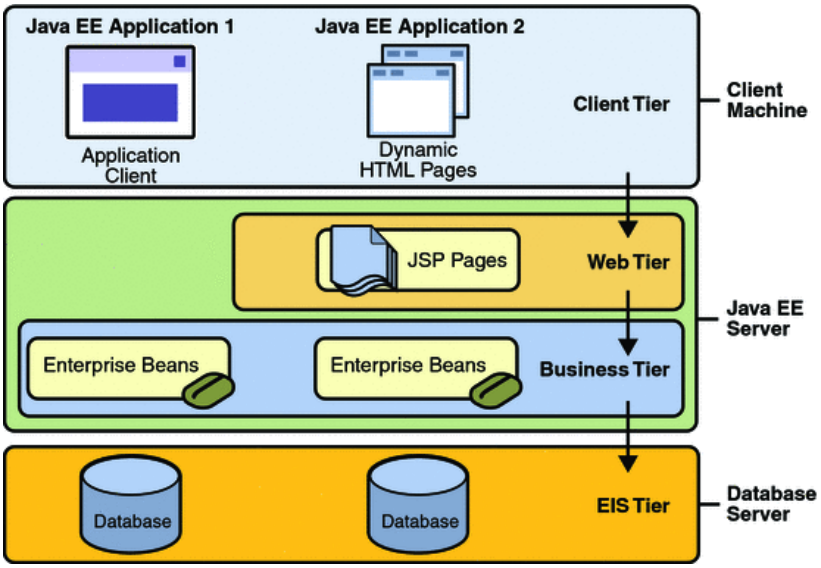

Figure 6: The Java EE Layered Architecture

Therefore, the EasyReach Cloud acts as a service container which exposes a set of "primitive" services that the clients can exploit in order to access the system functionalities. Figure 7 shows a functional view of the EasyReach Cloud architectural element. This view describes in more details the functional components that implement the social environment. The set of elements composing the EasyReach Cloud can be grouped into "public" and "private" components.

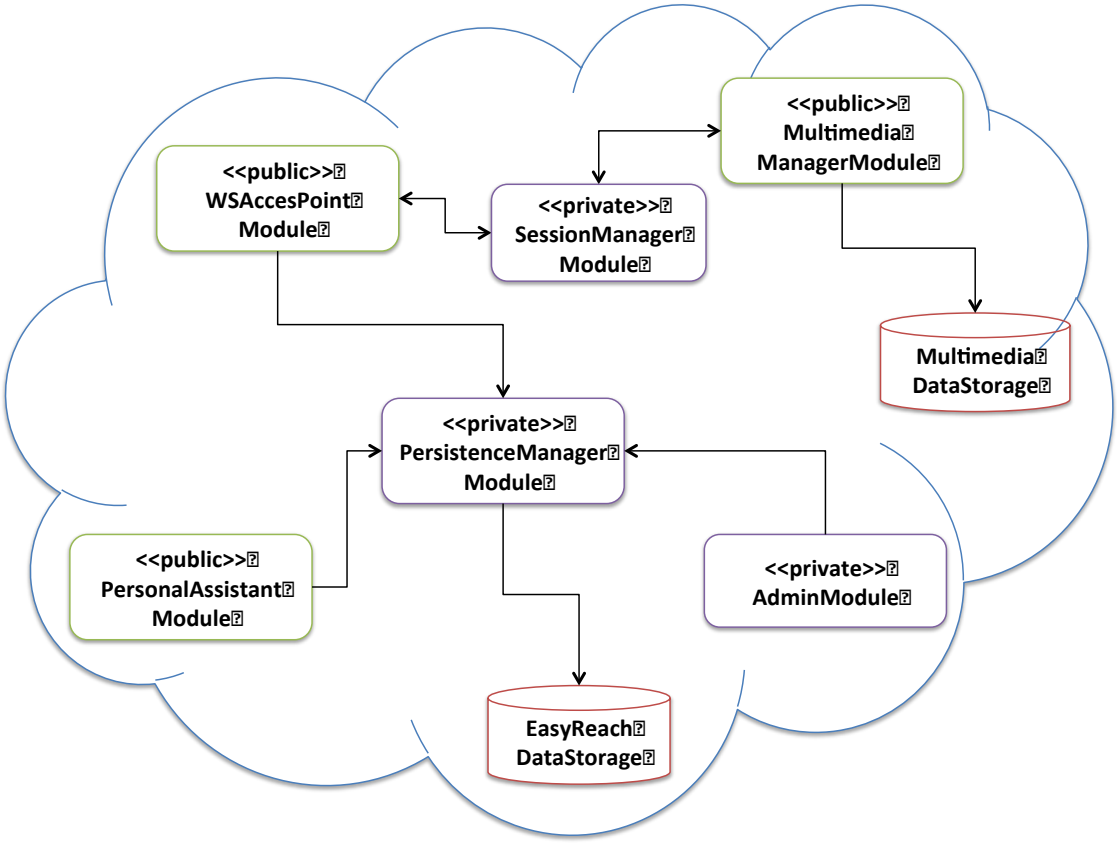

Figure 7: EasyReach Cloud Components

The subset of public components contains the services whose functionalities can be accessed from the outside of the EasyReach Cloud:

- The **WSAccessPointModule** is responsible for providing the clients with functionalities to access system information in order to realize the EasyReach Network. It is the "public" interface that defines a sort of "contract" through which the clients can interact and communicate with other users. From the outside of the EasyReach Cloud, it is possible to access System information only through the following services defined within the *WSAccessPointModule*:
    - *ElderActionService* – it defines the set of allowed operations that can be performed by users with "ELDER" profile.

- *GroupManagementService* – it defines the set of common operations needed for managing the EasyReach groups and the one-to-many communication among users within groups.
- *MessagingService* – it defines the set of common operations needed for managing one-to-one communication between users.
- *SystemService* – it defines operations for accessing common system information.

- The **MultimediaManagerModule** encapsulates a service dedicated to the exchange of multimedia files, i.e., audio, video and picture information (the *MultimediaManagerService*). This service exposes functionalities through which the clients can upload and download the multimedia content associated to the messages exchanged within the EasyReach system. Moreover, it is possible to notice from Figure 7 that the persistent storage of this kind of data is managed by a dedicated database that is different from the one used for storing EasyReach Network information.

- The **PersonalAssistantModule** encapsulates a special-purpose service (i.e., the *PersonalAssistanceService*) that provides the client-side *PersonalAssistant* (PA) element with read access to system taxonomy. It exposes a public interface by means of which PA modules can access system keywords and reason about users' interests in order to foster socialization suggesting new contacts and groups.

The private components subset instead, contains services that are only accessible within the EasyReach Cloud:

- The **SessionManagerModule** is responsible for managing users' sessions, by providing the *SessionManagerService*. Every time a user logs into the system, the SessionManagerModule generates a private key that uniquely identifies the user's session. Each private key generated is associated to a TTL (Time-To-Live) representing its time interval of validity. The key's TTL is renewed every time the associated user makes an activity into the system. If the TTL of a session expires (no activity of the user during the TTL) the private key is invalidated and a new authentication of the associated user is required in order to regain access to system functionalities.

- The **PersistenceManagerModule** abstracts the access to the persistence of system information, by means of the *PersistenceManagerService*. This service is responsible to directly communicate with the DBMS that manage the persistence of the EasyReach Network and exposes a set of functionalities that can be locally

exploited by other elements of the EasyReach Cloud in order to manage system information.

Therefore, the EasyReach system provides users with a set of high-level social services that result from the composition of "primitive" functionalities provided by the EasyReach Cloud in shape of Web Services. The client is responsible for composing the functionalities of the cloud in order to implement the desired social services exposed to the end users, as it will described in further details in the next sections.

3.3. The EasyReach Cloud: The EasyReach Data Storage

The EasyReach DataStorage is the element responsible for managing the persistent storage of system information. It has been realized by means of a several EJB controllers that interact with a MySql DBMS by means of JPA technology (Java Persistence API technology). Figure 8 shows the system information view, which describes the information managed by the EasyReach Data Storage and their relations.

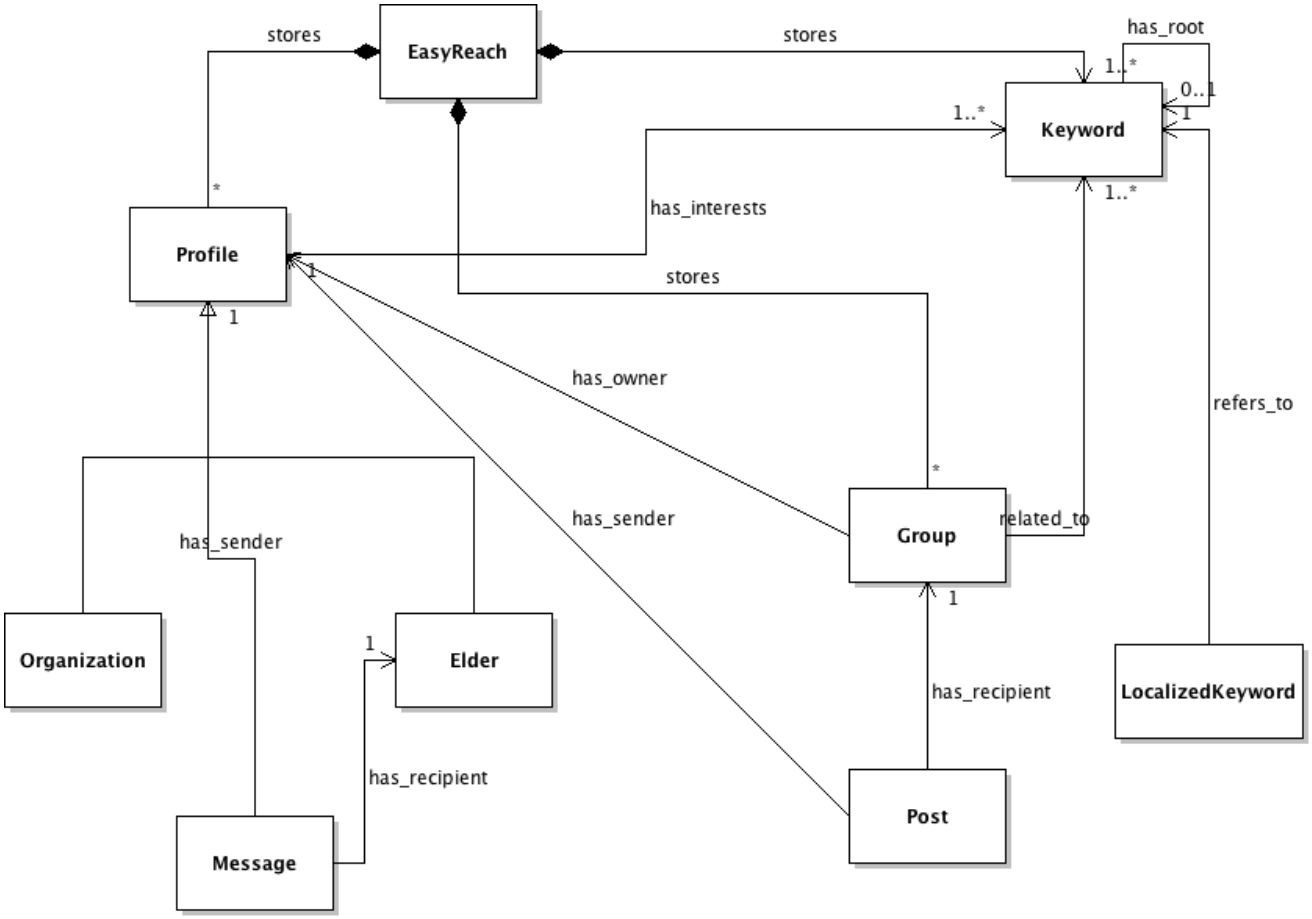

Figure 8: EasyReach Information View

The most important information managed by EasyReach are keywords, profiles and groups. A keyword is described by an unique ID, a unique tag, and a reference to a parent, which represents keyword's parent in the system taxonomy (root keywords have not parent). A profile represents the general information associated to system's users (user's ID, name, surname, password etc.). As Figure 8 shows, it is possible to store different kind of users (i.e., Elder and Organization). Each profile is associated with a set of keywords representing user's interests.
The system also stores information about groups. Similarly to profiles, each group is associated to a set of keywords representing group's topics. These are the most important structured information that the system exploits to build the high-level social functionalities that will further described in the next sections.

4. The EasyReach System base activities

This section is dedicated to the description of all the realized base activities that implement the basic EasyReach services, while in the next section, it will be explained how such basic services will be composed, and re-utilized, as building blocks to realize the high-level EasyReach Social Services. All the architecture hardware and software components presented in the previous sections cooperate to provide to the user an environment through which all the following set of activities can be comfortably executed:

I. Building contact lists
II. Subscription to existing groups
III. Creation of new groups
IV. Creation and sharing of messages to users and/or groups
V. Creation and management of a personal photo/video album
VI. Search for other users and groups
VII. Creation and submission of reminders to the personal agenda

As can be seen, the previous list of activities resembles very much the typical tasks performed within an ordinary social network. However, since the EasyReach system is designed to target an elderly users basin (possibly pre digital divide users) not familiar with technological devices and procedures, many efforts have been put to simplify the execution of the procedurs as much as possible, mainly (i) by keeping the technological aspects in the background, (ii) by limiting the quantity of information that the user must provide, and (iii) by re-utilizing the same system-user interaction templates used elsewhere throughout the system. In fact, today's technologies are too complicated and dispersive for people that are not accustomed to them (let's think, for instance, to social application on smartphone, designed for a small display with a lot of "crowded" icons, or

even the web, a really dispersive place due to its unstructured nature and the huge quantity of information within it).

Following the previous guidelines, the designers have therefore organized the system-user interaction protocols as composed of a small number of simple actions and patterns, trying to re-utilize the same patterns (both logical and graphical) whenever needed, ultimately creating for the user a graphical user interface which might help keeping the system utilization learning curve as steep as possible (i.e., the steeper the learning curve, the less experience is necessary to learn, the faster is the learning process).

Figure 9: The EasyReach Social Environment

Figure 9 shows the main EasyReach GUI page that appears immediately after the user's login. We can identify two main sections on the screen:

I. *The "Frame"*, representing the static part of the environment, composed by the scrollable contact bar (at the bottom), that contains the user's contacts (friends, relatives, as well as the ones proactively suggested by the system), the scrollable group bar (at the left), where the user can find her/his selected thematic groups (and the suggested ones), and the command bar (at the right), containing the list of the actions that the user can perform within the system (e.g. taking a picture/video, creating a group and so on)

II. *The Information Area, i.e.,* the central part of the screen is the dynamic section, which changes on the base of the user's actions and/or selections: for instance, if the user selects a contact from the bottom bar or a thematic group from the left bar, the Information Area will show the messages exchanged between the

user and the selected element. On the other hand, if the user visits his gallery or wishes to create a group, the Information Area will switch to the proper screen, therefore providing a structured and simple way to present information to the user.

In the following sections, after presenting each base service activity, we will delve more deeply into the description of how the elements described in Section 2 work together towards the realization of the same activities, by focusing on the internal relations among the system elements, with the help of UML interaction diagrams.

4.1. Building Contact and Group Lists

The *Contact List* (Figure 9, bar at the bottom) is the place where all the user's contacts are stored and visualized, and is created as follows. Every time the user logs into the System, a special software module called *Personal Assistant* (see Deliverable 3.3 for further information) is launched to proactively select a specific number of possible contacts on the basis of the personal interest that they may share with the user, and with whom the user might be interested to get in contact. Such automatically selected contacts will therefore be visualized in the contacts bar of the interface. However, the user is always entitled to directly add to this list any contact of her/his choice, regardless of whether they have been selected by the Personal Assistant or not. In this case, the system will visualize in the Contacts bar also the contacts actively chosen by the user. In fact, the user can ignore the system's suggestions at any time, and build up her/his own *Contact List*, by adding new users by performing a *Search Operation* (see Section 4.6).

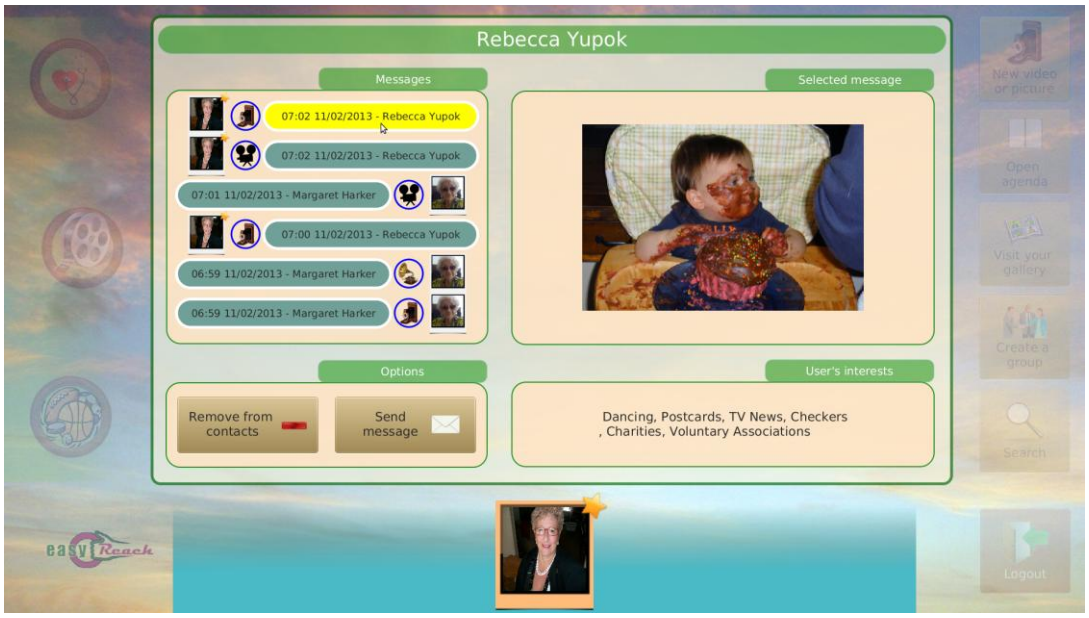

Figure 10: A user's personal contact

Moreover, the user can act on the contact list by further personalizing it. The *Personal Contact List* is a subset of the whole contacts set visualized on the interface, composed of all suggested contacts which are *approved as interesting* by the user. After user's approval, those contacts will appear as marked with a star symbol on top of their related icons (see the leftmost contact on Figure 9), and will become *Personal contacts*. As opposed to ordinary contacts, all personal contacts will always be visualized in the *Contact List*, every time the user logs in future sessions; in fact, the contacts not marked as *Personal* are generally not persistent, in that the Personal Assistant cannot guarantee that they will be selected for visualization in future sessions, as they might be replaced by other contacts that exhibit more interests in common with the user. However, the user retains full control on the contacts management, as she/he can always add or remove a contact to her/his *Personal Contact List* by clicking on the special button "Add to/Remove from contacts" (see Figure 10). The contacts that are possibly removed do not receive any notifications from these operations. The Group List building (Figure 9, the bar on the left side of the screen) follows the same procedure described for the Contact List. It contains all groups already subscribed by the user and some new groups automatically suggested by the System.

4.1.1. User Access and Contact List creation interaction sequence

This complex functionality is obtained by the collaboration of all software architectural elements described in the previous sections. Figure 11 shows an UML Interaction Diagram, which describes how these elements work together in order to implement this functionality. As it is possible to see, all the information provided in the main view are computed at login time. Indeed, when a registered user logs into the system, the *SocialEnvironment* element calls the `login()` method of the *ModelController* element (see Figure 3). First of all, the ModelController sends an authentication request to the *ElderActionService* by means of a login message. If the user is successfully authenticated, the *SessionManagerModule* (see Section 3.2) creates a new session with a private key in order to identify the user.

Once authentication is successfully performed, the *ModelController* initializes the Personal Assistant (PA) in order to get suggestions for the current user. The PA exploits the *PersonalAssistantService* in order to read system taxonomy but also the users' and groups' interests. Then the PA reasons about these information and provides the *ModelController* with the set of groups and contacts the user is likely to best appreciate in her/his contact list in order to foster socialization. Finally the *ModelController* initializes listeners that run in background in order to notify the user when new messages or posts are received. At the end of these operations the *SocialEnvironment* element is ready to display the System main screen to the user, shown in Figure 9.

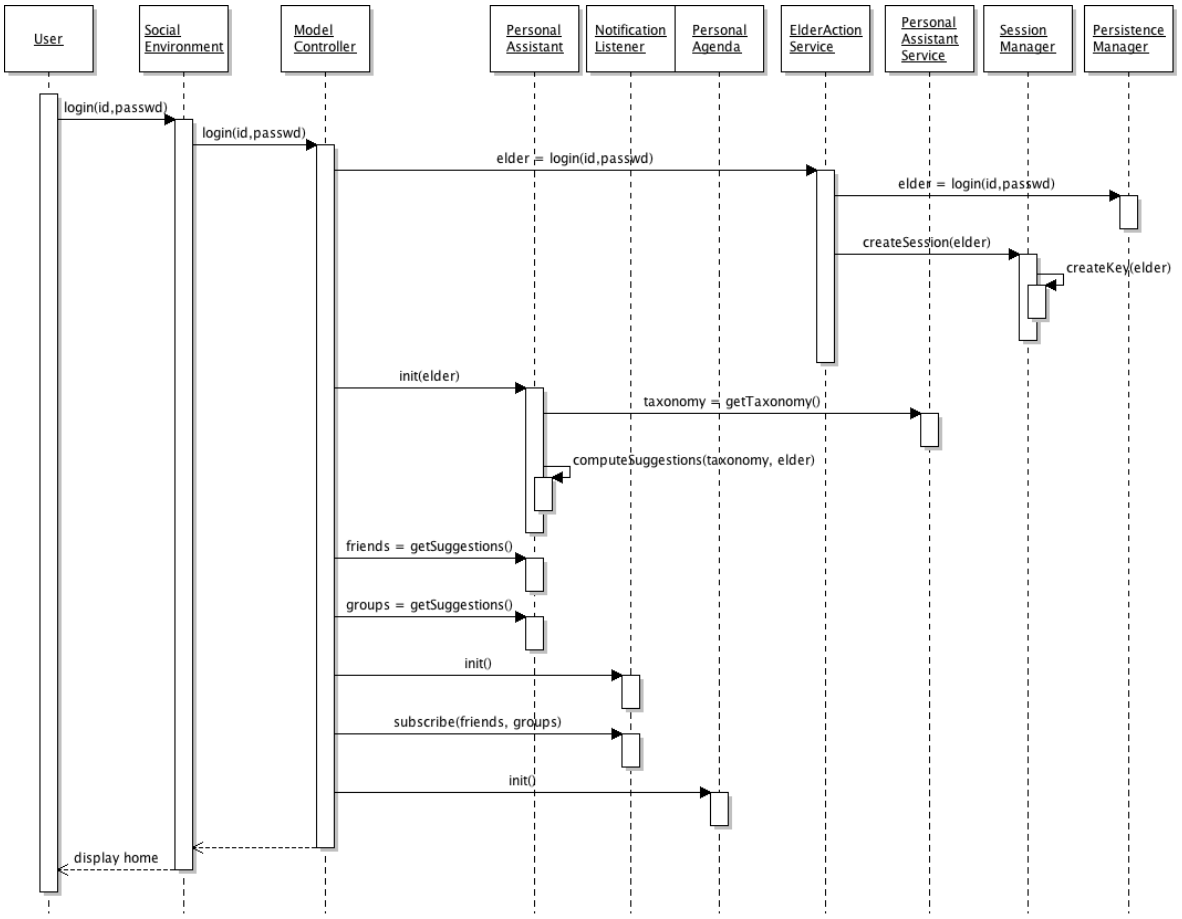

Figure 11: The user's access and contact list creation UML Interaction Diagram

4.2. Subscriptions to existing Groups

Likewise to the *Creation of contact lists* (Section 4.1), after login the user is presented with a list of groups of interest that can be customized by subscribing/unsubscribing to/from them (Thematic Group bar on the left, see Figure 9). A group represents a personalized discussion place where all the users in the system can post multimedia contents for sharing information, experiences, etc. Each time the user logs into the System, the Personal Assistant module suggests some groups to the user, reasoning on both (i) the possible matching between the user's interests and the group's topics and arguments, and (ii) how frequently the user has entered the discussion group in the past sessions (which is considered as a sign of possible interest). Following a policy identical to that used for the contact lists, the user's subscription to a group entails that such group will always be shown in future sessions (the group will be visually marked "subscribed" with a star on its related icon); the same is not guaranteed for any other unmarked group.

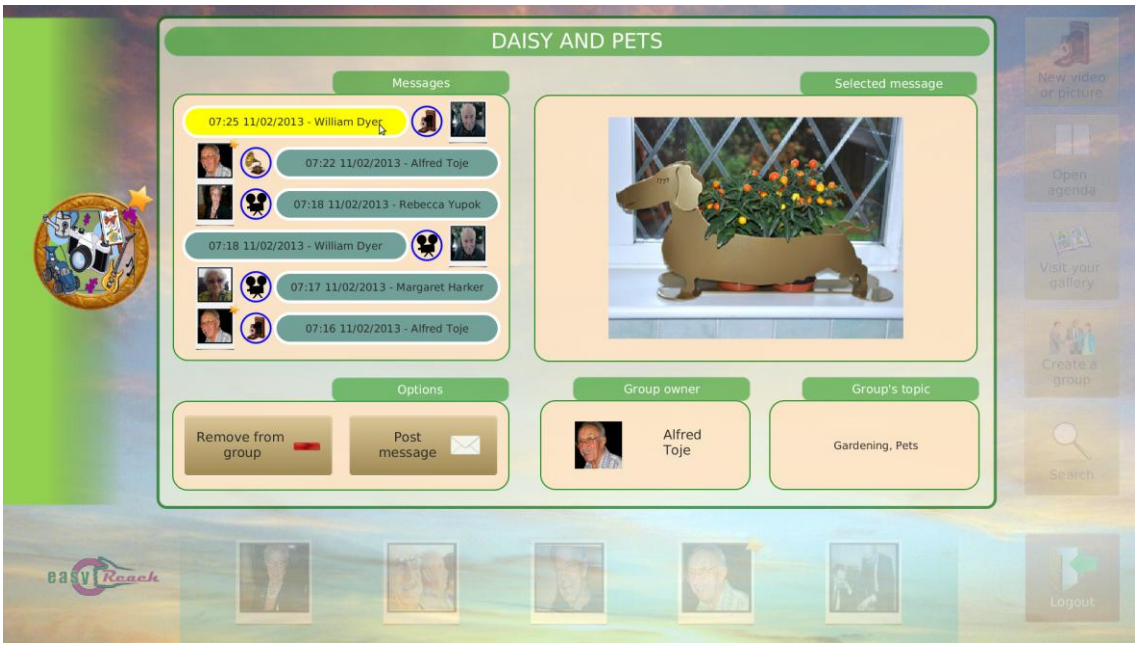

Figure 12: A group message exchange page

Figure 12 shows an example of group message exchange page that is presented to the user as soon as she/he selects a group from the thematic group bar. The star icon appearing on the selected group's icon represents the fact that the group is among the user's preferred groups (i.e., user has already subscribed to the group).

Again, as for the *Contact List,* and regardless of any selection performed by the Personal Assistant, the user can add any groups to the *Group List* through a *Search Operation* (see Section 4.6). Also, the user can create a new group by using one of the services offered by the system and described in the next Section 4.3.

4.2.1. Group Subscription interaction sequence

A user can subscribe any group he is interested to, by clicking on the "Subscribe" button into the related group view, as described above. Figure 13 shows an UML Interaction Diagram, which describes the interactions occurring among the system internal elements in order to realize this functionality.

When a user wants to subscribe to a group in the group bar, first he has to access to the group information view. When the user selects the group, the *SocialEnvironment* element receives the request for changing the current view. Then, it calls the `getExchangedPosts()` method of the *ModelController* in order to load all group related posts. The *ModelController* can fetch this data locally in its cache if already downloaded, or remotely by calling the *GroupManagerService*. The *GropuManagerService* checks the user session by means of the *SessionManager* and then reads the data from the *PersistenceManagerService* to assess whether the session is still valid. After these operations, the group information view is shown to the user.

Subsequently, when the user clicks on the “Subscribe” button, a subscription request is sent to the *SocialEnvironment* element, which in its turn calls the `subscribe()` method of the *ModelController*.

The *ModelController* forwards the request to the *ElderActionService*, which stores this information by means of the *PersistenceManagerService*. If the subscription is successfully executed, the *ModelController* finally notifies the PersonalAssistant about the user’s activity and adds the subscribed group to the *NotificationListener* in order to keep the user updated when new posts are published into the group.

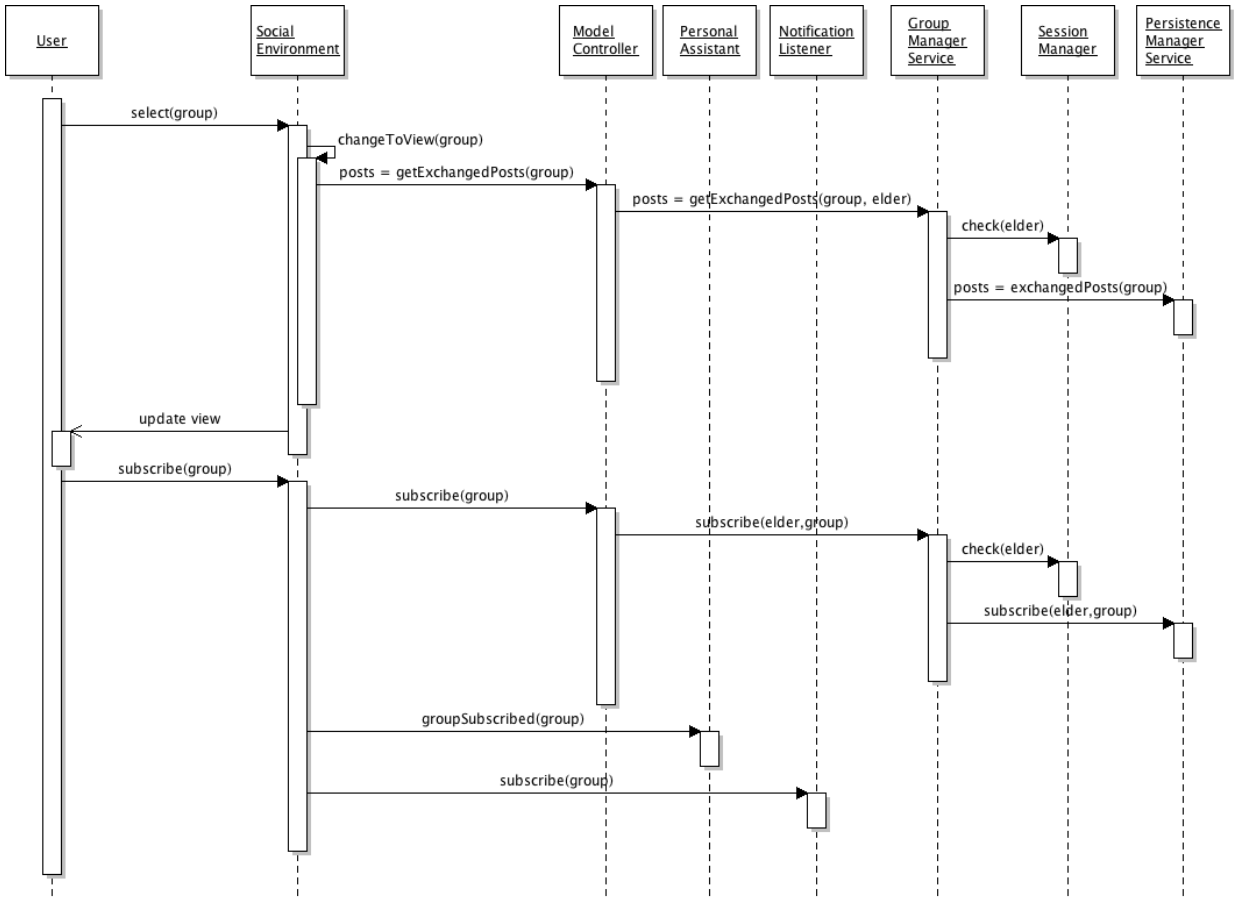

Figure 13: Group subscription UML Interaction Diagram

4.3. Creation of new Groups

By exploiting the *“Create a group”* service the user can create a new thematic group in the System. The service can be launched by selecting the “Create a Group” button from the main page (see Figure 9), and the EasyReach system provides the user with a simplified screen for group creation (see Figure 14). Once displayed, in order to create a new group, the user must enter the group’s main line of interest and the specific topics

associated to that line, as well as the group's name. The user can also choose if the group that she/he wishes to create is a public one (visible to everyone) or a private one (visible only to the contacts currently present in his *Personal Contact List*). Once all the required information is provided, the group creation can be finalized by selecting the "Create Group" button.

Figure 14: Creating a new group

4.3.1. New Thematic Group creation interaction sequence

A user can create groups on certain topics he likes. In this way a user can create a communication channel by means of which he can easily share experience and knowledge with other users that share the same interests. Figure 15 shows an UML Interaction Diagram, which describes the interactions needed among architectural elements involved for implementing this functionality.

From the main view (see Figure 9) the user accesses the Group creation view (see Figure 14). At that point, the *SocialEnvironment* element calls the `getKeywords()` method of the *ModelController*, which sends the request to the *SystemService* in order to fetch the system taxonomy keywords. Once in the Group creation view, the user can assign a name to the group he/she wants to create and can select the set of keywords representing group interests (see Figure 14). When the group creation is confirmed by the user (the group owner) by clicking on "Create group" button, the *SocialEnvironment* element calls the `createGroup()` method of the *ModelController*. Consequently, the *ModelController* registers the new group into the system through the *GroupManagementService* so that the group will be available to other users for

subscription. Once the group has been successfully created, the *ModelController* adds the new group to the *NotificationListener* element, in order to receive notifications when posts are published into the group, and updates the *PersonalAssistant* about the user's activity by invoking `groupCreated()` method.

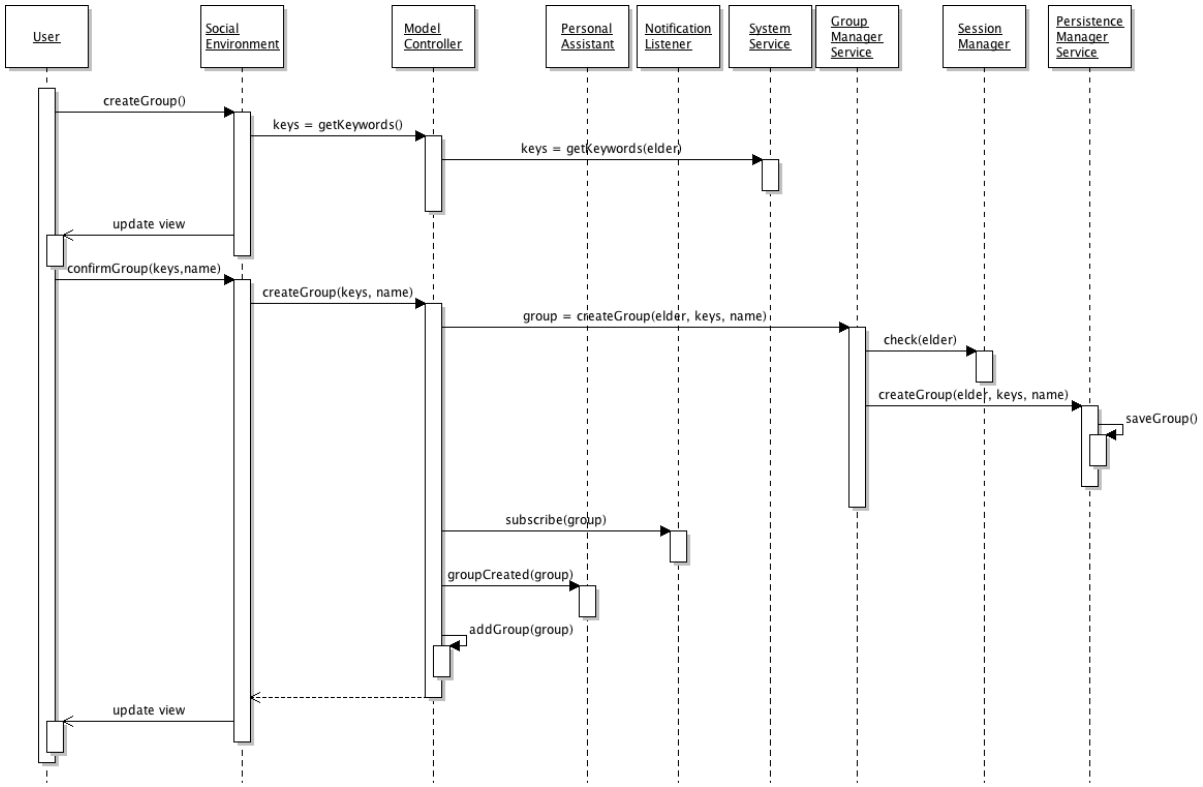

Figure 15: The group creation UML interaction diagram

4.4. Creation and sharing of messages to Users and/or Groups

As described in Section 2.3, the *Remote Control* offers the possibility to capture multimedia contents (like pictures, video and audio file) that can be shared with other users or groups in the system. Figure 16 shows the *Send new message* screen the EasyReach user is presented with when she/he is about to send a multimedia message to one of her/his contacts. This screen can be reached by clicking on the "*Send/Post a message*" button present in every profile in the System, either for contacts and for groups (see for example Figures 10 and 12, respectively).

In particular, the example of Figure 16 refers to the operation of sending a picture file to a single contact. As clearly shown in the screen, after accessing the *Send new message* page the user has the choice of either (i) selecting the multimedia content to be sent among the ones previously saved in her/his own personal library (see the *Choose Video* and the *Choose picture* boxes in Figure 16), or (ii) capturing a new content to send at the

same time she/he is creating the new message (the creation and management of the personal multimedia album will be described in the next Section).

All received messages are temporarily stored in a specific folder located in the STB (and managed transparently to the user) and will remain there for the entire duration of the current session. As the user logs out, all the contents of this local folder are deleted in order to save hard disk memory; all the locally deleted messages will be re-downloaded on request from the EasyReach server in future sessions.

Figure 16: Send a new message

4.4.1. New media capturing interaction sequence

A user can gather new multimedia content at any moment using the Remote Control. Figure 17 shows the UML Interaction Diagram describing the interactions among the system elements. When a user clicks on the “Take a picture” button, or on the “Record audio” or “Record a video” buttons, sends a request to the *SocialEnvironment* element. The *SocialEnvironment* communicates to the *RemoteControl* to set itself in “take picture” mode through the *SocialController* and update the user view. Then, when the user clicks on the “OK” button of the RemoteControl, the current image is acquired and it is sent to the *SocialEnvironment*. Finally, the new image is displayed to the user, which can save it to local gallery or send it directly to the user or group.

All the files stored locally into the STB can be seen from the user by accessing his personal gallery through the System. In general, user’s gallery files are uploaded to the server only when they are shared after sending messages or posts.

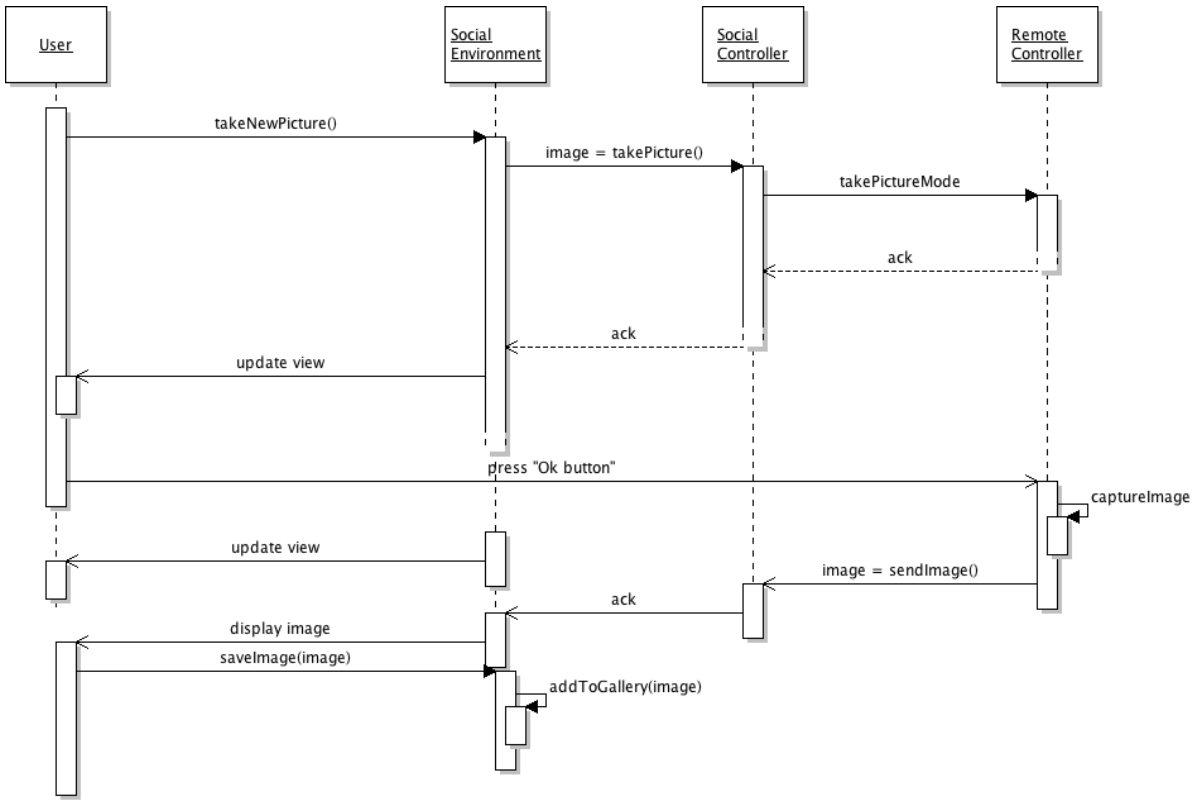

Figure 17: The media capturing UML interaction diagram

4.5. Creation and management of a personal photo/video album

During the system's utilization, the user can at any time capture new multimedia contents (pictures, audio and video files) through the *Remote Control* and store them in a special location called *User Gallery.* By selecting the "*New Video or Picture*" button (see Figure 9), the user enters a page from which it is possible to create new multimedia content (see Figure 18). The management of this page is straightforward: by selecting either the "*Record a video*" or the "*Take a picture*" button, the user is immediately prompted to use the RC to record new contents. Independently from the selected content type, a preview of the captured element is shown in the page, so that the user can decide whether to keep it or discard it. In the first case, by selecting the "*Save element*" button, the content is finally saved in the user gallery, which can be visited at any time, as described in the following.

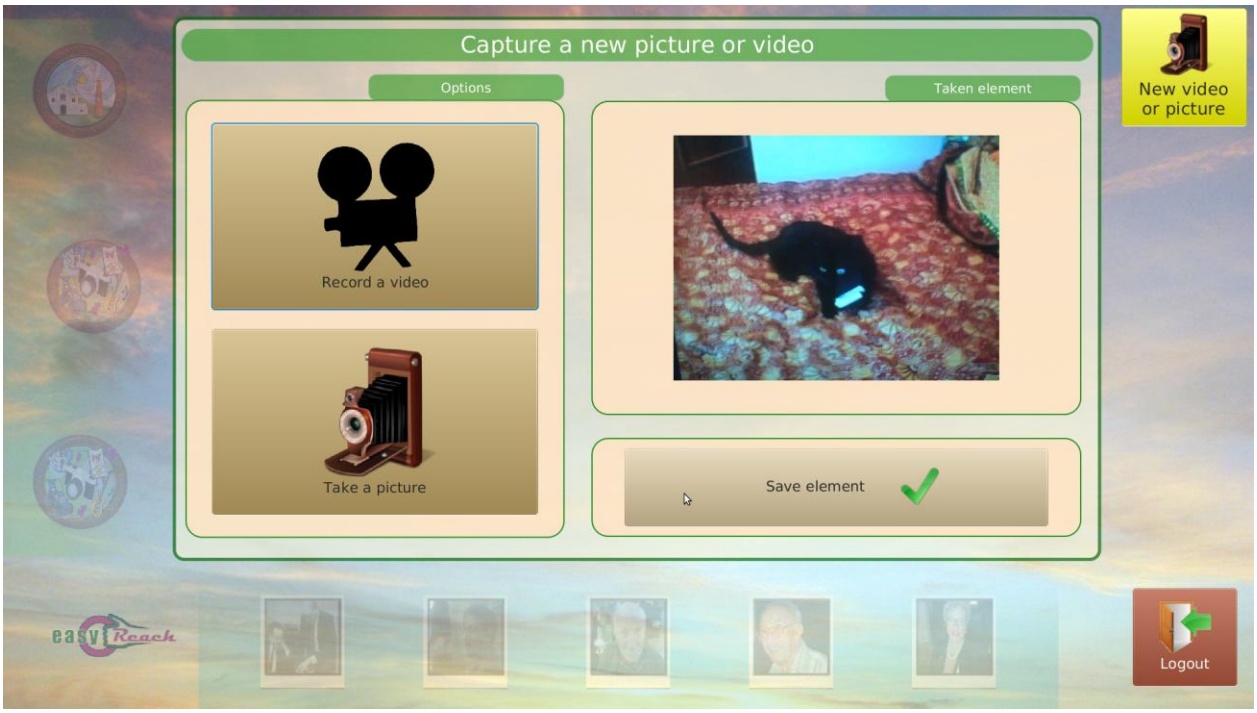

Figure 18: EasyReach page accessed to create new multimedia content

The EasyReach graphical environment provides the user with an immediate access point to perform the access to the user's personal gallery of images and videos, by means of the "*User Gallery*" button (see Figure 9). Every time this button is selected, the user is re-directed to the user gallery page, an example of which is presented in Figure 19; this page acts like a sort of album where the user can save all the pictures and video possibly acquired in the past, and/or browse through them at her/his best preference. As anticipated in the previous Section 4.4, the User Gallery can be used to select the multimedia contents to be sent to other contacts and/or groups; in other words, the user is not forced to create the message's content at the time of message sending.

From the technical standpoint, in order to minimize the volume of data exchanged in the network, the *User Gallery* has been designed to store all the information locally in the STB, meaning that all the files are saved in the local memory and are never sent to the central server, unless they are selected by the user as part of an outgoing message.

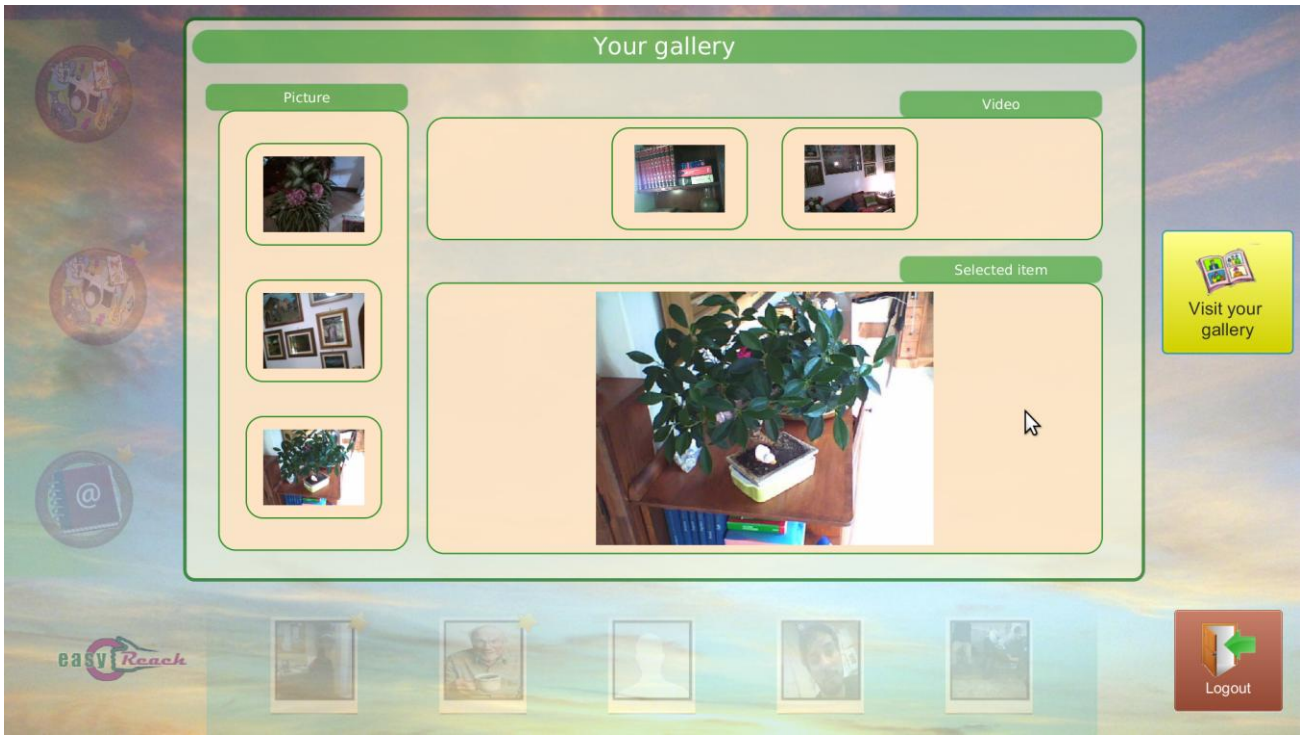

Figure 19: User Gallery

4.5.1. Gallery file selection and Message sending interaction diagram

The System allows users to easily capture new media by means of the RemoteControl, as described in the previous section. Then users can share these data by selecting them from their personal gallery and sending messages to friends or posts to groups. Figure 20 shows the UML Interaction Diagram describing the System internal elements involved and their interactions when sending a message to a contact.

Initially, the user selects a contact from his “Contact Bar” in the main view. The *SocialEnvironment* element calls the *ModelController* in order to load all the exchanged messages between the user and the selected contact, and then it displays the related contact view. In more detail, the *ModelController* retrieves the exchanged messages by means of the *MessagingService,* if it does not found them in its local cache. Subsequently, the user is allowed to click on the “Send Message” button (see for example Figure 10), and the *SocialEnvironment* calls the `loadGallery()` method of the *ModelController* and updates the view with the videos and images loaded from the gallery (see Figure 16). At this point, the user selects an image to share and confirms the operation, triggering the *SocialEnvironment* to call the `sendMessage()` method of the *ModelController,* specifying the recipient of the message and the image selected by user.

At this point, the *ModelController* element uploads the selected image to the server by calling the *MultimediaManagerService*; once the media has been successfully uploaded,

the message is actually sent to the recipient by means of the *MessagingService* functionalities (or *GroupMenagementService,* when sending a post to a group). Finally, the *MessagingService* element calls the *SessionManager* to identify the user, and then sends the request to the *PersistenceManager* in order to actually save the message into the System.

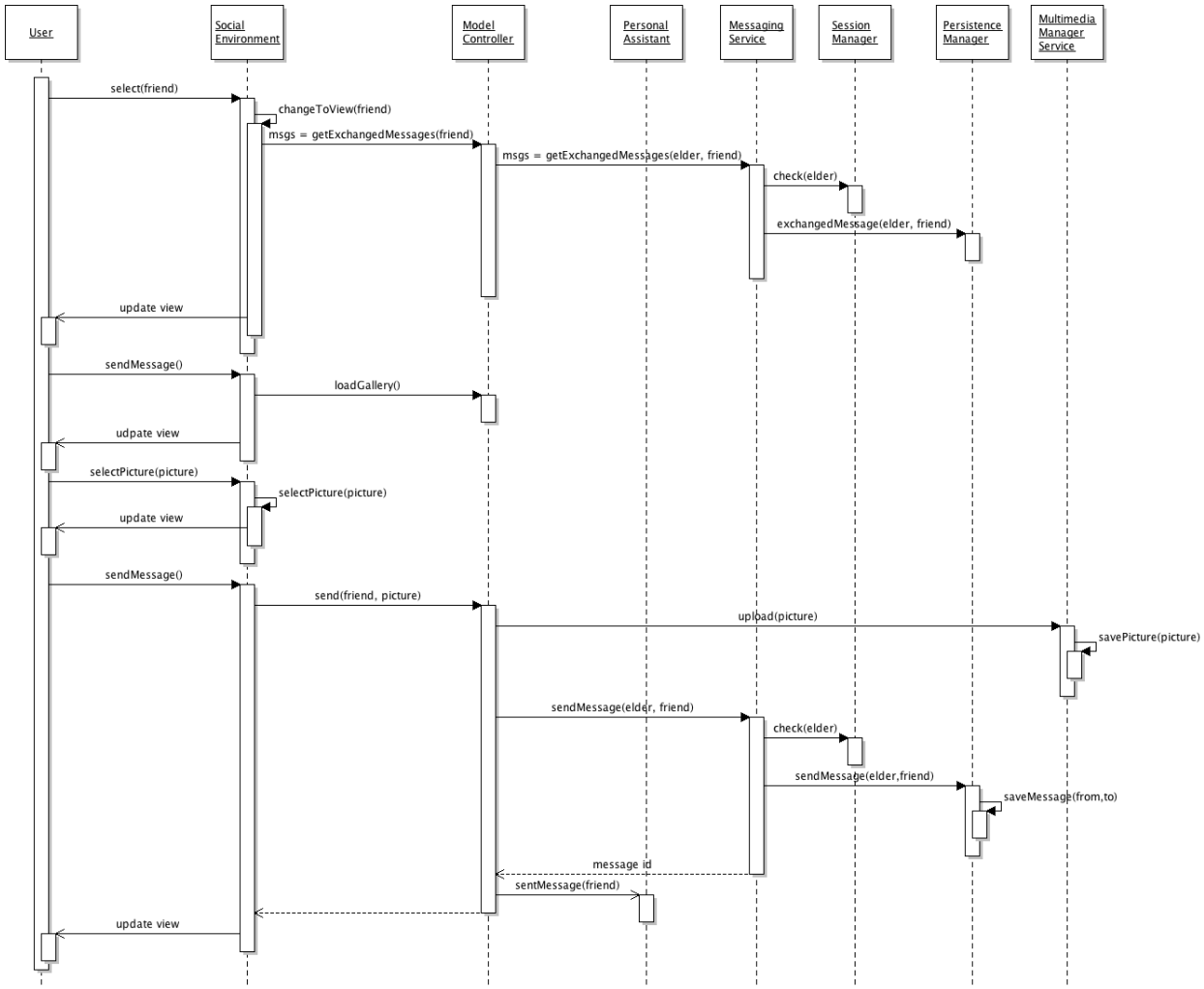

Figure 20: Media selection and message sending UML interaction diagram

4.6. Search for other Users and Groups

As explained in Section 4.1, the initial contact list and the initial thematic group list are automatically generated by the system's Personal Assistant on the basis of the interests that the user has with respect to the other contacts' interests or groups' discussion topics. But what if the user should like to add to her/his list one or more contacts that are not selected by the Personal Assistant (e.g., other Easyreach users whom she/he knows personally but who do not share interests with her/him)? In all cases where the user is

interested in contacting a specific user or group that are not being suggested by the system, she/he can use the *Search Service* in order to add these elements to the *Contact list* or to the *Group List* (see Figure 9).

When the user accesses the *Search* service, the screen shown in Figure 21 is presented; as the figure shows, in order to search an item in the EasyReach cloud, the user must provide a set of minimal information, such as the name (even partial) of the element to search, and the kind of search specification (user search or group search). This choice was motivated by the need to minimize and simplify as much as possible the data to be provided by the user. Once all information has been inserted, the user can finally click on the Search button. Figure 21 shows an example of search results; after the search process has completed, the system presents the user with a list of all the profiles that match the search criteria, even if partially submitted, leaving the user with the only task of eventually accessing the desired profile by simply clicking on one of the results, and directly start any conversation or join a discussion on a group.

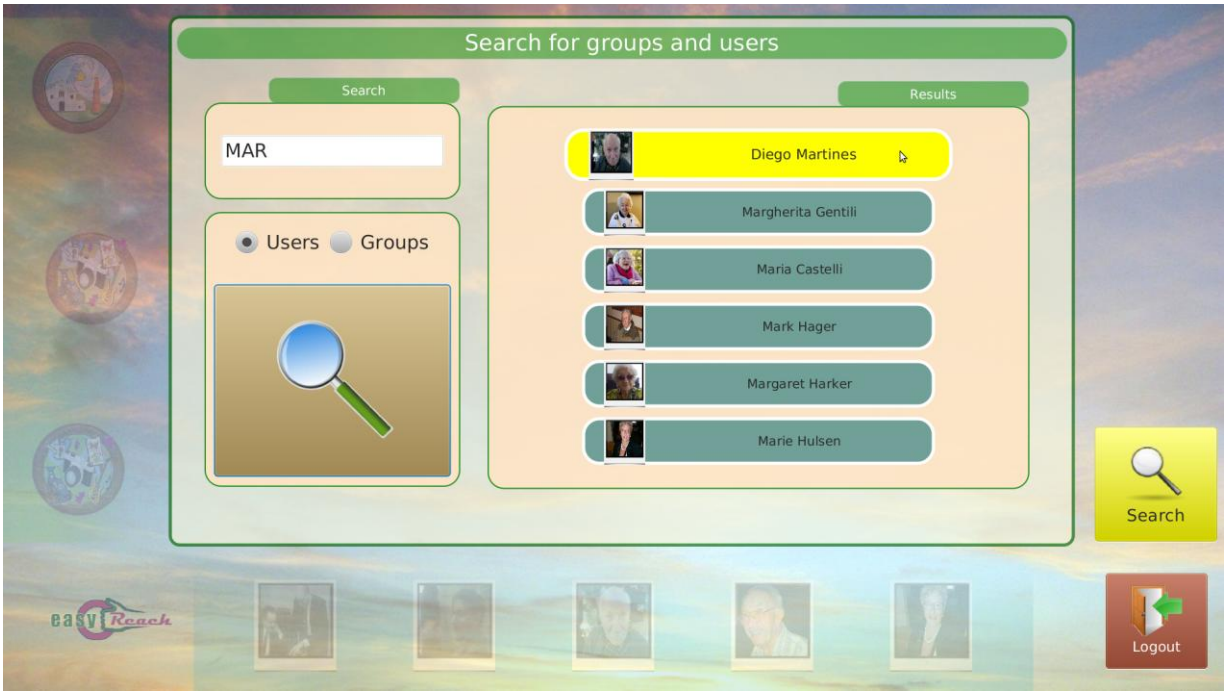

Figure 21: The Search Service

4.6.1. Users and Group Search Interaction diagram

A user can look for a specific person not automatically suggested by the system. Figure 22 shows the UML Interaction Diagram, which describes how this functionality is actually realized with respect to the elements composing the Systema architecture.

Initially, the user inserts a textual query by means of the “Search view” (see Figure 21) and selects the type of search he wants to perform (either user or group search). The *SocialEnvironment* element receives the search request and calls the *ModelController*

element using the `search()` method. Then, the *ModelController* sends the query to the *ElderActionService*, which returns the set of users (or groups) matching the input query. If the user finds the desired item, he can select it in order to access the related information view (user or group view). The bottom half of the diagram shows how the item is added to the contact or group bar of the *SocialEnvironment,* similarly to what described more thoroughly in Section 4.2. In this phase, no information is sent to the *PersonalAssistant*.until the item is added as user's friend or subscribed group.

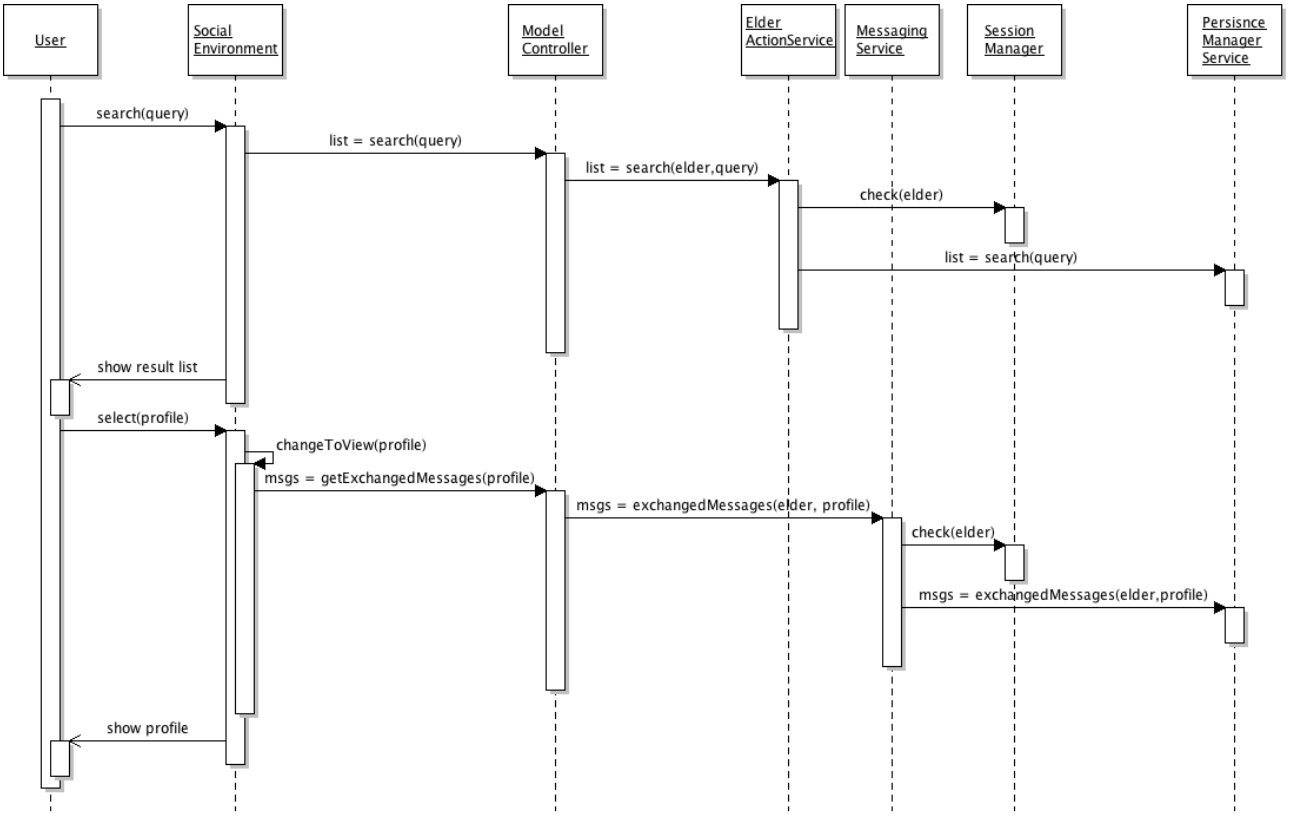

Figure 22: User and Groups search UML Interaction Diagram

4.7. Creation and submission of reminders to the Personal Agenda

Among the services that EasyReach offers to the elderly users, the Agenda service is particularly useful as it provides proactive support for the user's daily life, by generating on-line notifications relatively to the user's personal commitments. The main objective of the EasyReach Agenda is to remind important deadlines and/or managing appointments by providing to the user a comfortable and direct environment through which to access and provide all the necessary information about the reminder messages she/he wishes to be notified in the future. In other words, through the agenda service, the user has the possibility to send timed messages to herself, i.e., by deciding what to communicate (the contents of the message) and when to communicate it (i.e., the time of the notification).

Informally, the Agenda service works by providing a continuous monitoring of all the data input by the user, constantly checking their "expiration date" against the system's clock time, and "firing" the proper notification event when the two coincide. This section is dedicated to the description of the procedures for creating and submitting new reminders to the Agenda.

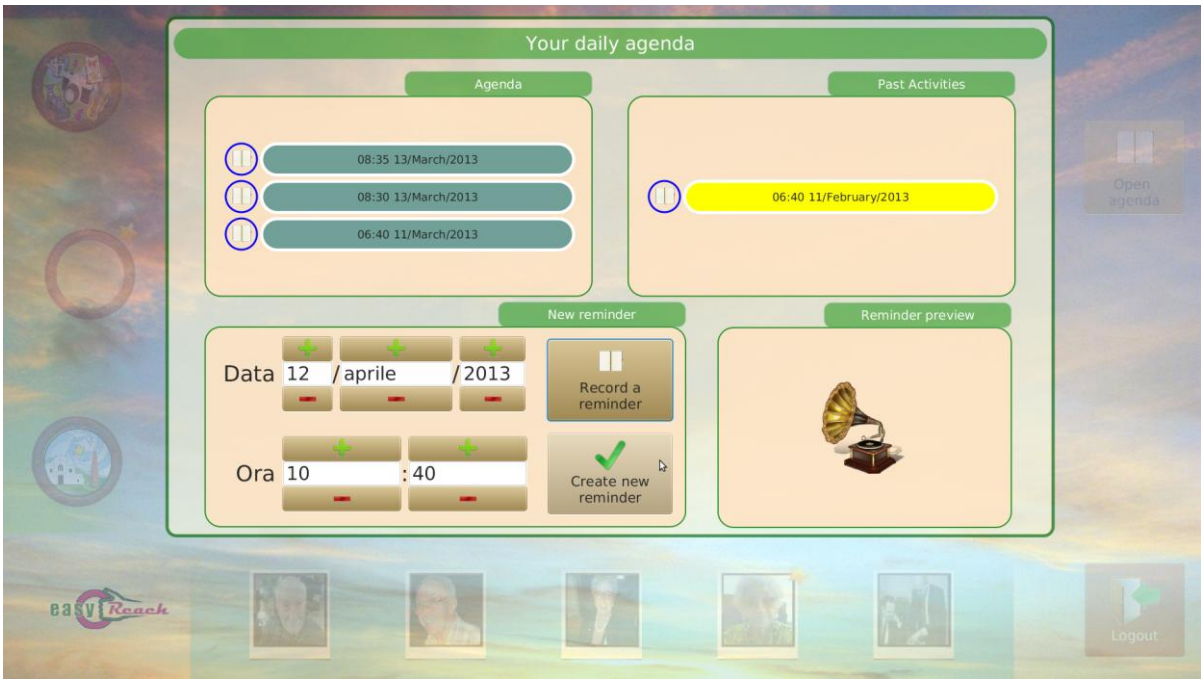

Figure 23: The EasyReach Agenda screen

The user accesses the agenda service by activating the "*Open Agenda*" command button (see Figure 9). By pressing the button, the information area of the screen changes to that shown in Figure 23. The environment presented to the user does not offer many differences relatively to the ordinary contact or group message exchange screen. The list of all received notifications (i.e., the messages) is still present at the top-left of the view in the "*Agenda*" box, and all messages are presented in the familiar fashion. The same holds for the "*Reminder preview*" box (bottom-right), which can be used to enjoy a preview of the newly created notification before it is added to the agenda. The "*Past Activities*" box (top-right) contains all the past notifications that have already been notified to the user.

In order to submit a new reminder to the agenda, the user is requested to provide two types of information: (i) the data (i.e., date and time) about when the reminding will have to be notified to the user, and (ii) the contents of the reminding itself. The date and the time of the notification can be provided in a straightforward fashion by acting on the "*Date*" and "*Time*" button arrows, (day-month-year, and hour-minutes, respectively), while the contents of the reminder can be recorded by the user as an ordinary audio file, by pressing the "*Record a reminder*" button. It should be underscored that, in order to facilitate the

utilization of the agenda, the creation of a reminder audio file follows exactly the same procedure as the creation of any other audio message to be sent to one of the user's contacts or groups (see Section 4.4). Finally, once the reminder is created, its contents can be listened back by clicking on the "*Reminder preview*" box, before definitely adding the message to the agenda by means of the "*Create new reminder*" button.

4.7.1. Daily Agenda Reminder Creation interaction sequence

When the user accesses the "Agenda View", he can record new reminders which will be automatically notified by the system in due time. Figure 24 shows the UML Interaction Diagram describing the required interactions among the system elements.

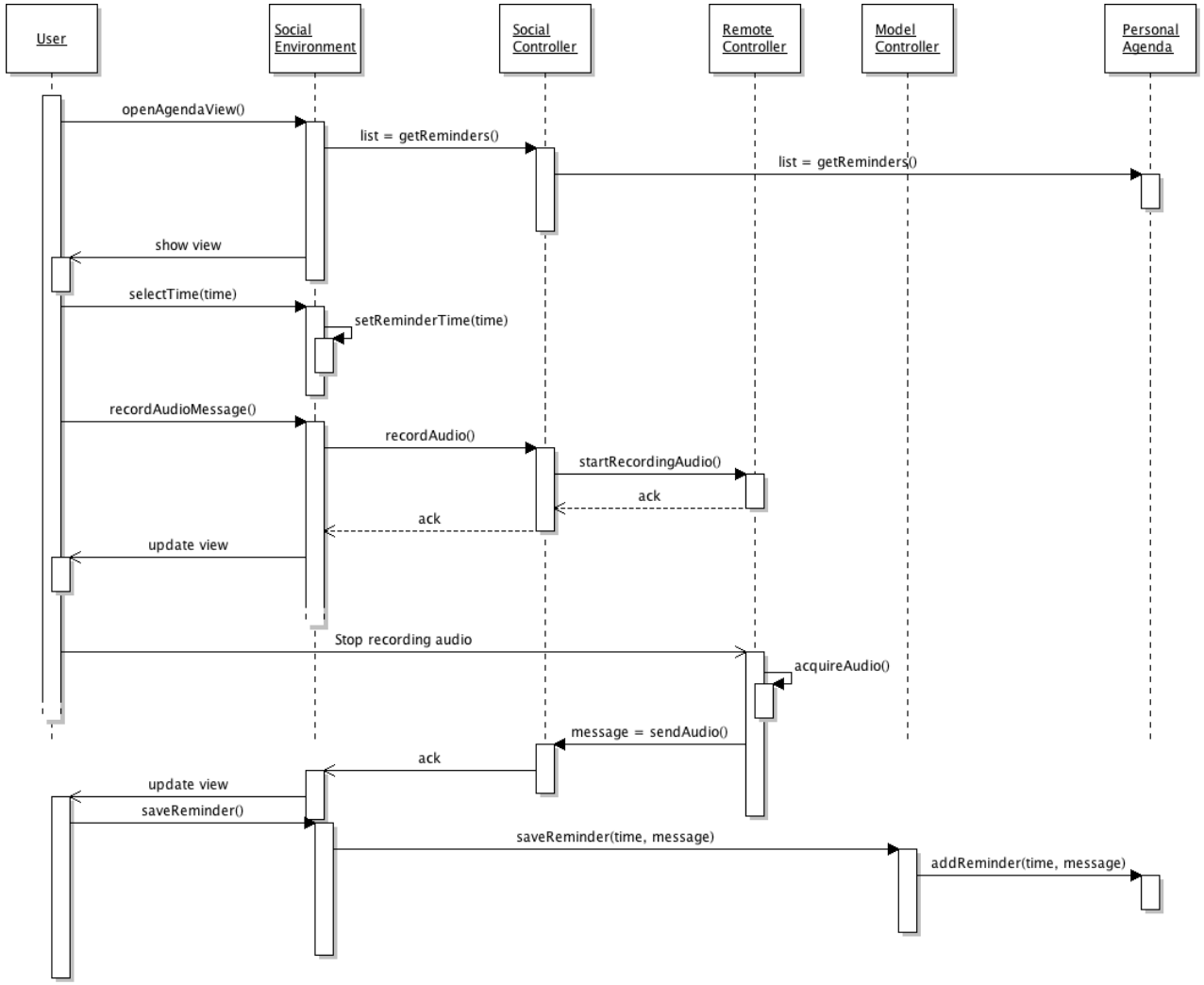

Figure 24: The Agenda Reminder Creation UMl Interaction Diagram

After the agenda view is displayed, the user selects the time at which he/she wants to be notified. Then, the user registers an audio message related to the event to notify by

means of the *RemoteControl,* as already described in Section 4.5. When the user is satisfied about the produced reminder, the method `saveReminder()` of the *ModelController* is called in order to propagate the request to the *PersonalAgenda* element, which is ultimately in charge of managing all saved reminders, and issue the related notifications.

5. The EasyReach Social Services and Examples

The base services presented in the previous section serve the purpose of creating by composition a set of high-level social interaction tasks (also called Social Services) that represent the ultimate objective of the project, i.e., the realization of specialized versions of social network activities, specifically tailored for elderly people. It is worth to underscore once more that the main design guidelines for the implementation of such services has been the simplicity of system learning and utilization, basically obtained through the re-utilization of known patterns. In other words, as the realization of the base services presented in Section 4 has followed the rationale of re-using as much as possible elementary procedural and graphical patterns in order to minimize the complexity possibly perceived by the user, in the same way, the social services described in this section are built by exploiting the previous base services, whenever required.

The social services that have been implemented in the current version of the EasyReach system are the following:

- Organizing sets of contacts, including relatives and friends
- Creating groups of people that care for a certain topic
- Organizing “interface” groups with existing organizations
- Organizing help sessions where a skilled user can help or train other users

5.1. Organizing sets of contacts, including relatives and friends

The examples listed above are representatives of typical needs that the system must support in order to fulfil the overall project objectives. The main objective of this social service is to allow the user to create and manage a circle of contacts with which to interact and share personal contents. For instance, any elderly user constrained at home for a long period of time might feel the need to keep in touch with her/his friends of the local senior center, with some relatives, and/or with other EasyReach users that share her/his own interests.

Let us suppose that the user would like to use the system to exchange some messages with some other users in the EasyReach cloud. Initially, the system relieves the user from the burden of searching other contacts to possibly get in contact with, by

preparing a preliminary list of possible contacts on the user's behalf. As the user logs in, she/he is immediately put in the condition to select one or more contacts and exchange with them personal contents related to the selected contact's interests. This feature is of great help to encourage communication success and reduce the feeling of isolation, as the probability of establishing an enjoyable communication is proportional to the extent of the overlapping between the peers' respective preferences. For example, an elderly user who is very interested in needlecraft, will most likely find a certain number of other users interested in the same discipline already in her/his contact list, ready to be contacted, and the same holds for any other hobby or interest. The user can choose one of such pre-selected contacts and visit the relative page; visiting the page does not necessarily entails selecting the contact as "favourite" (i.e., making the contact stable in the contact list), but rather it might be useful to get a first grasp of the contact's other interests beyond the one she/he has been searched for. Whenever the user decides, it is always possible to send a greeting message to the new contact, in order to present oneself; at the same time, upon receiving the message, the recipient user can immediately check the sender's interests, and decide whether to start the "conversation" or to postpone it. Eventually, both users can decide to add each other among the selected contacts, in order to guarantee their constant presence in each other's contact list and therefore facilitate future communications.

Besides the contacts automatically filtered by the Personal Assistant, the generic EasyReach user will certainly want to add to her/his contact list a number of relatives and/or friends. We can easily imagine situations where the user wants to share the pictures of her/his grandsons, or videos of their last birthday, to some friends. In order to add a specific EasyReach contact to one's list, it is sufficient to know her/his name (or part of it), start the contact search as explained in Section 4.6 and, once the contact is found, immediately select it as a favourite by marking it with a star, as explained in Section 4.1. This last operation ensures that the contact will be found in the user's list in any future login sessions.

In the following, we will show how the back-end and the logic behind the social environment allow the user to interact and use the services that the EasyReach system offers to him/her, relatively to the specific case at hand. In order to explain all the functionalities, we will start from this subsection, describing a possible session of an EasyReach user starting from the login operation. Likewise, the same "walkthrough" organization will be followed in all the following subsections.

The first action that the user performs is the *Login* (see Figure 25), in order to join the EasyReach social channel. To do that, the user has to select his/her profile from the ones loaded on the STB (a STB shared among many users, e.g., a senior center's STB, could have more than one profile saved on it); subsequently, the user enters a visual password by selecting a number of predefined icons in the correct order.

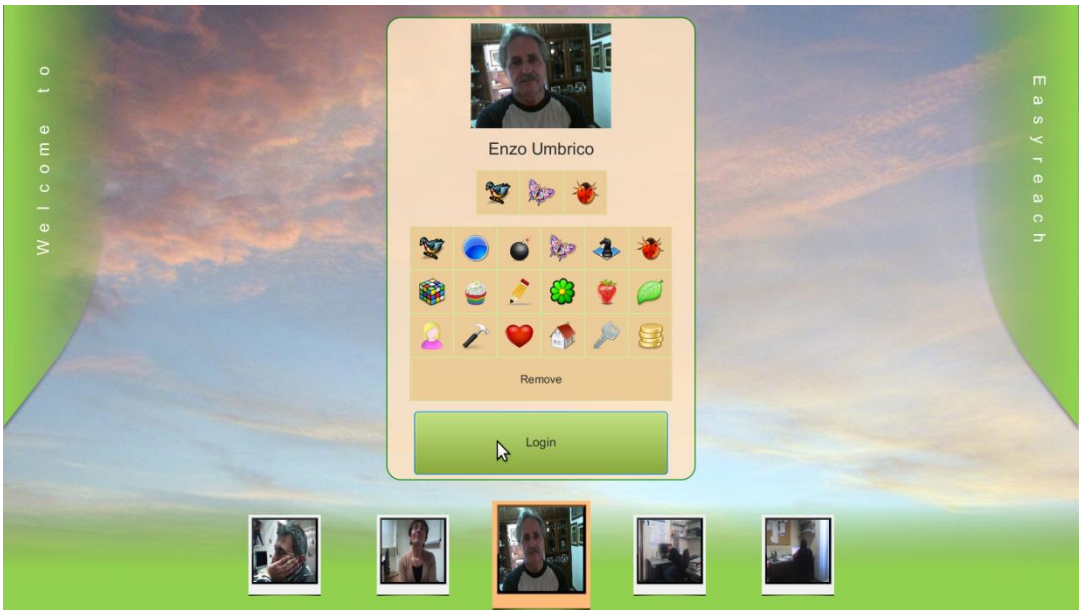

Figure 25: User Login

When the user clicks on the Login button, the system will call the remote service for checking if the entered credentials are correct. If the server validates the username and password, the user is permitted to join the system. After the login, the server will send to the STB-client all the information about the user's contacts and groups; in fact, by following the Model View Controller (MVC) pattern (shown in figure 26), all the information will be sent to a special element in the Back-End layer called *Controller* (see Figure 3). This object takes charge of the responsibility to notify the Social Environment of any update from the server (and vice versa).

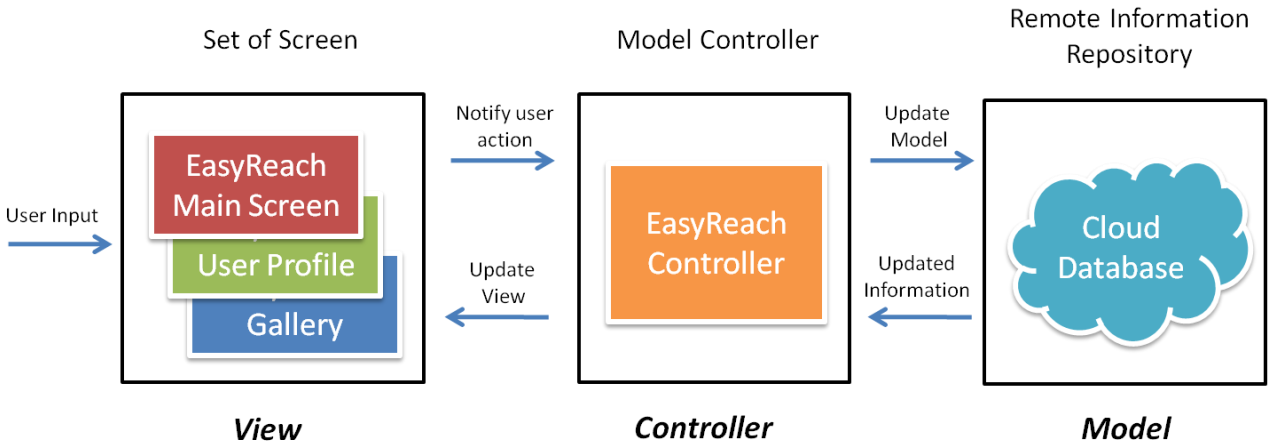

Figure 26: MVC Pattern Sketch

Therefore, once the user is successfully logged into the system, the *ModelController* is in charge of automatically retrieving all the information about the contacts and groups the

user cares about. In this way, the *ModelController* builds the user's personal view of the EasyReach Network making accessible the user's contacts and subscribed groups. Moreover, the *ModelController* calls the *PersonalAssistant*, which accesses the system taxonomy and reasons about the user's interests in order to compute new users and groups (suggestions) the user would probably like to join. All the interactions required among the system elements involved in the creation of this "personal view" of the system have already been described in the UML Interaction Diagram of Figure 11 in section 4.1 (login operation).

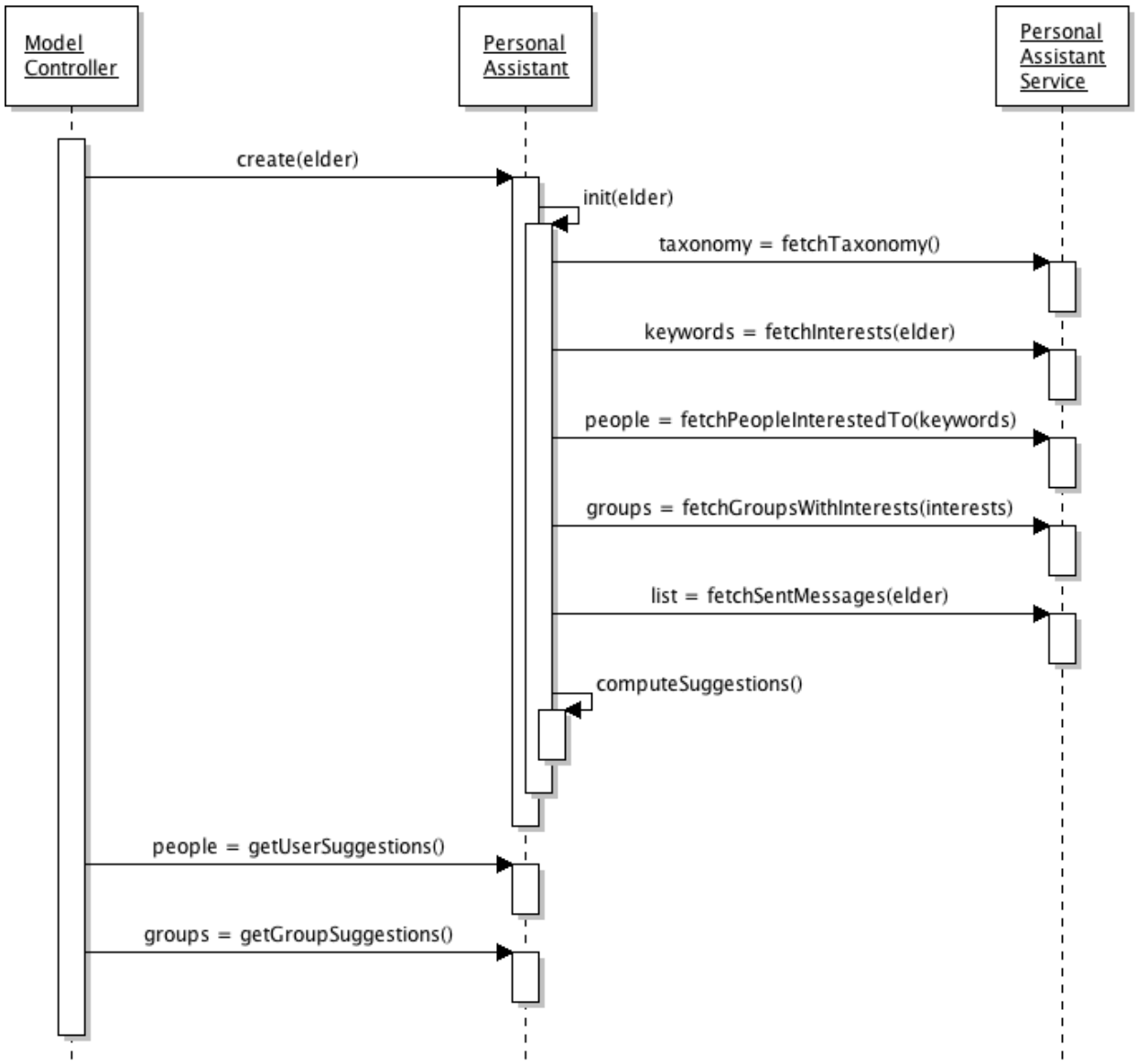

Figure 27: *PersonalAssistant* creation and initialization

However in Figure 27, another UML Interaction Diagram is used to further describe the interactions between the *ModelController* and the *PersonalAssistant* in order to explain how these elements interact for fostering socialization among users.

How it is possible to see in such diagram, the *PersonalAssistant* is initialized by passing user's information (the parameter named `elder` in the `init()` method). The *PersonalAssistant* collects all information required to compute suggestions for the considered user through the *PersonalAssistantService*. The *PersonalAssistant* fetches the set of keywords and relations that describe the system taxonomy by calling the `fetchTaxonomy()` method exposed by the *PersonalAssistantService*. Then, the *PersonalAssistant* identifies the set of keywords representing the user's interests and uses them to filter people and groups from the system, that share same interests with the user by means of the `fetchPeopleInterestedToKeywords()` and `fetchGroupWithInteres()` methods. Subsequently, it calls the `fetchSentMessages()` method to analyze the user communication activity in order to assess other possible connections that do not simply depend on keywords in common. Once the *PersonalAssistant* has collected all these information, it is ready to compute suggestions by means of the `computeSuggestions()` method. Finally, when the login operation is complete, the *ModelController* sends all collected information to the *SocialEnvironment*, which will draw the graphical elements that contain the information on the user's TV, as shown in Figure 28.

At the bottom part of the screen, the user can find all the contacts that have been selected by the personal assistant, by reasoning on the common interests, and the friends added by the user himself (tagged with a star). The same occurs on the left part of the screen, relatively to the group list. On the right part of the screen, all the services offered by the system are statically shown as clickable buttons (e.g., take new pictures, manage the agenda, etc.).

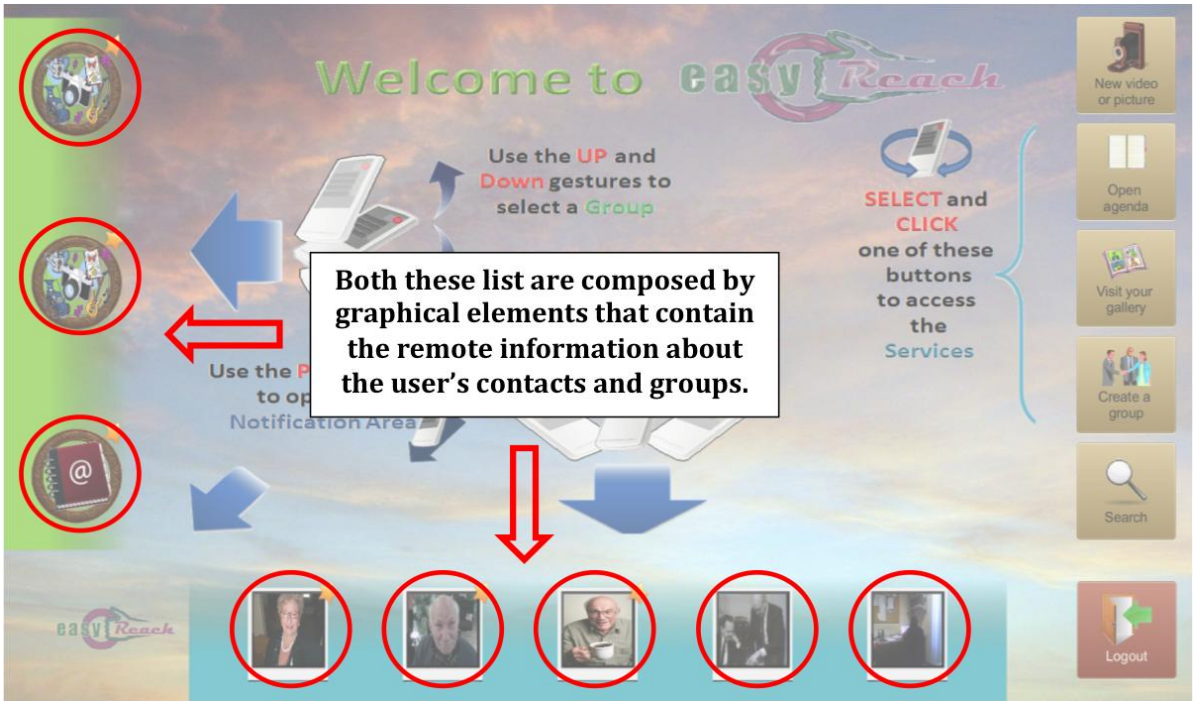

Figure 28: Main screen

Let's now suppose that the user would like to add a specific person to its contact list, for instance, one of his/her friends, and let's assume that the Personal Assistant has not selected the desired profile to the user's contact list.

Figure 29: The Search Functionality

In this case, the user has to search manually for the chosen profile, by using the Search Command on the right side of the screen, as shown in Figure 29. After selecting the Search Command, the Social Environment will update the system's view by showing the Search Screen (Figure 30).

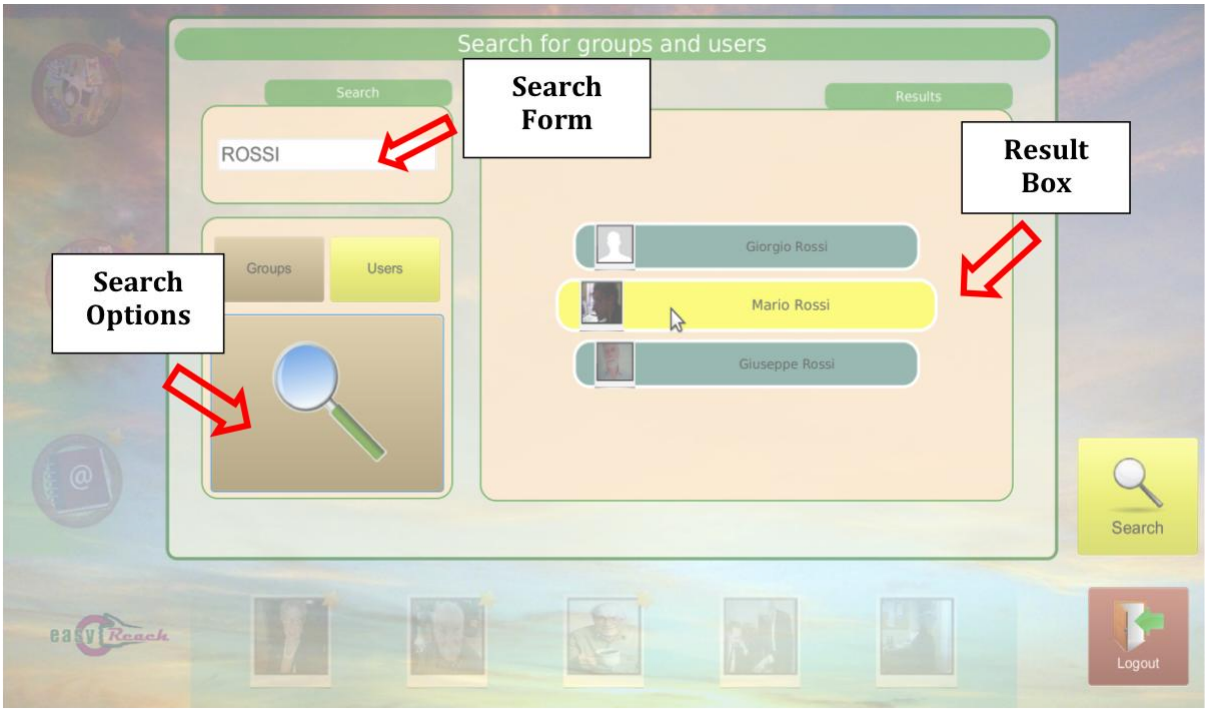

Figure 30: The Search Screen

Here, the user can insert the (partial) name of the person he is looking for in the search form on the upper left of the screen. In order to do that, the system will show a virtual keyboard when the Search form is selected (shown in Figure 31).

Figure 31: EasyReach Virtual Keyboard

For instance, let's suppose that the user is looking for a friend called Mario Rossi. The Search Form is filled with the partial string "Rossi" and the *User* button is selected (i.e., to search for a user instead than a group). To start the search, the user clicks on the magnifier button and a remote request for all the profiles with the string "Rossi" in the name/surname is sent to the remote server. Eventually, the server will send back all the requested information, and the social environment will show the results in the Result Box on the right part of the screen. Then, the user can select the desired profile by clicking on it and the system will open the Conversation Screen (Figure 32).

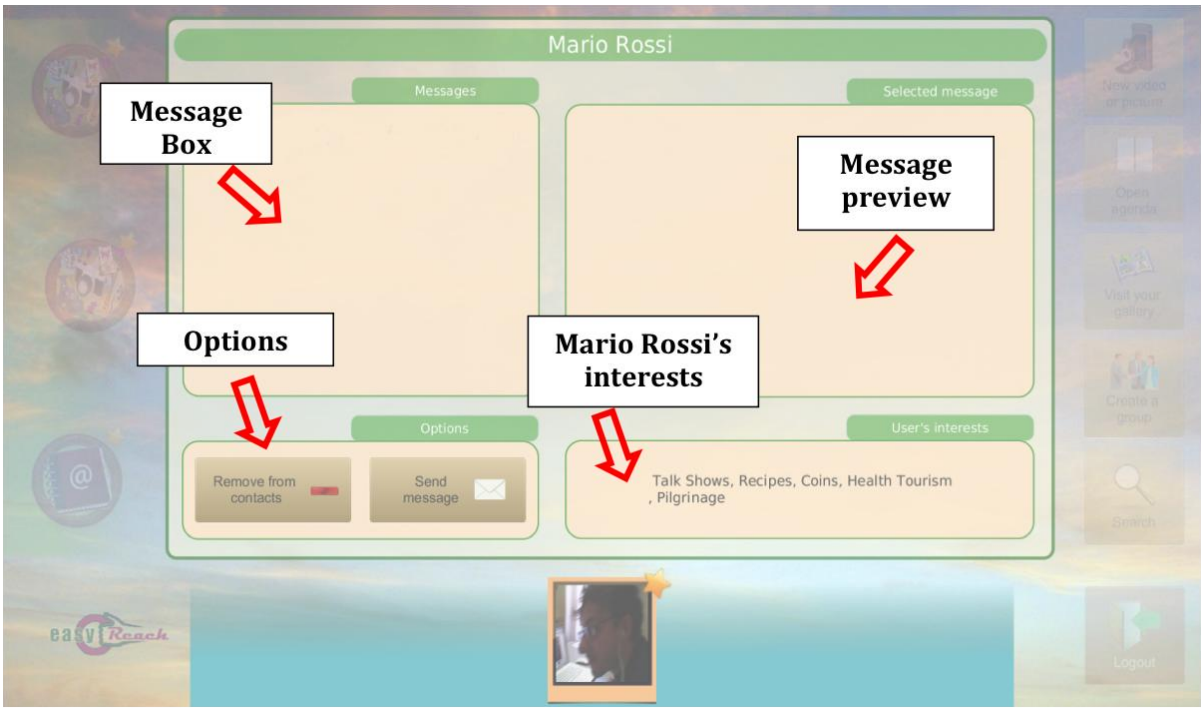

Figure 32: A Conversation Screen

The Conversation Screen is the section of the social channel that allows the exchanging of messages between two users. The screen is divided into four areas:

- *Messages Box*: all the messages exchanged are visualized here. In the example, the user has just added the profile to its contact list, so the Message Box is empty.

- *Message Preview*: here the system will show a thumbnail of a specific selected message.

- *Options*: this box hosts two buttons: the left one allows the user to add or remove the current profile to/from its personal contact list (i.e., "Add to contacts" and "remove from contacts, respectively"), the right one is used for sending a message to the selected contact.

- *User's interests*: here the user can find the contact's main interests. This is the information used by the personal assistant for reasoning and suggesting new contacts.

Continuing with our example, the user can add Mario Rossi in its personal list by clicking on the left button of the *Options* box. After this, the contact will be tagged with a star and the same button will be switched to a "Remove from contacts" button.

Figure 33 shows the UML Interaction Diagram describing the interactions among the system elements when a user adds a contact, as described above. When the user clicks on the "Add to contacts" button, the *SocialEnvironment* receives ad `addFriendship()` request, which is forwarded to the *ModelController*. The *ModelController* calls the `addFriendship()` method of the *ElderActionService*, which checks the user session through the *SessionManager* and stores the request into the system database by calling the `addFriendship()` method of the *PersistenceManagerService*. Once the information are stored into the system, the *ModelController* notifies the *PersonalAssistant* of the new connection by calling the `notifyFriendhisp()` method.

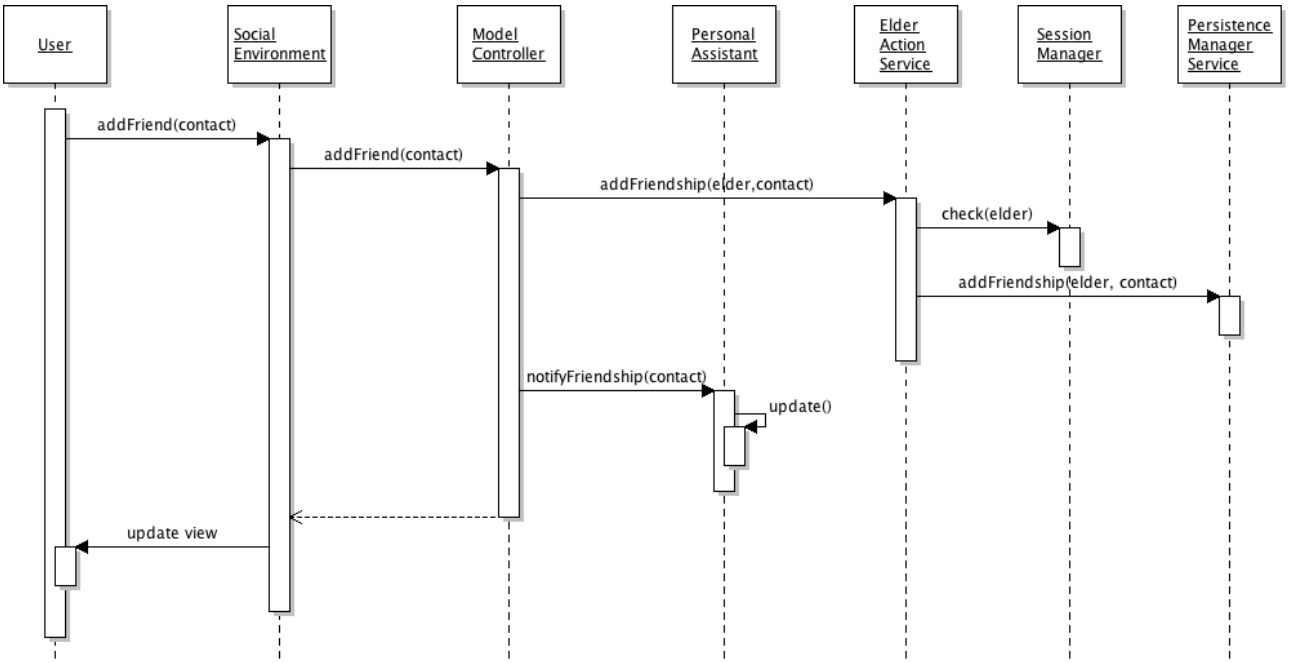

Figure 33: Add a Friend operation

Now, let's imagine that the user would like to send a message to its new contact: in order to do this, he clicks on the "Send Message" button, and the screen shown in Figure 34 will be visualized. In this screen, the user can select a multimedia content that could be sent to the contact: the system offers the possibility to send an old content (a previous

saved picture or video) or to create a new one. In this case the user selects an old picture (its preview is displayed in the central box) but, in case he would like to capture a new picture, or record a new audio/video, he can do this by clicking on one of the buttons on the right box (“Create new msg”).

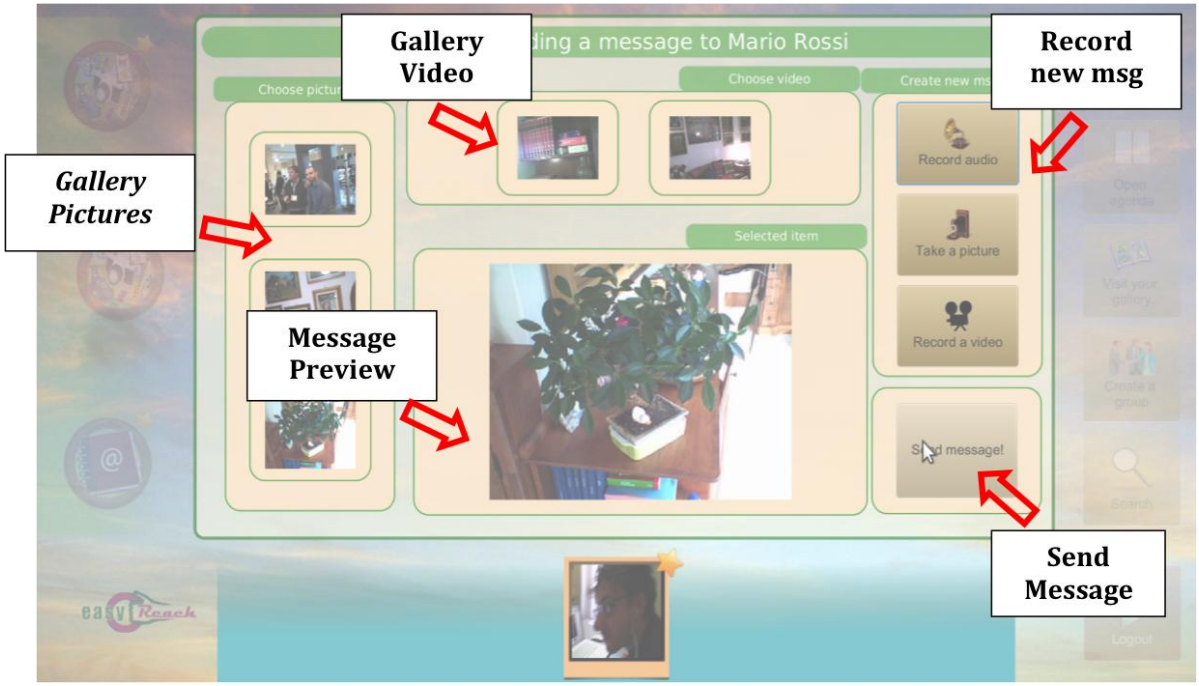

Figure 34: Sending a message to Mario Rossi

In this last case, the system will notify the user’s intention to the Remote Control Server, that will set the Remote Control’s Status in Capture Mode. When in Capture Mode, the OK button on the remote control will act as a trigger for taking a new picture or starting the recording of a new audio/video.

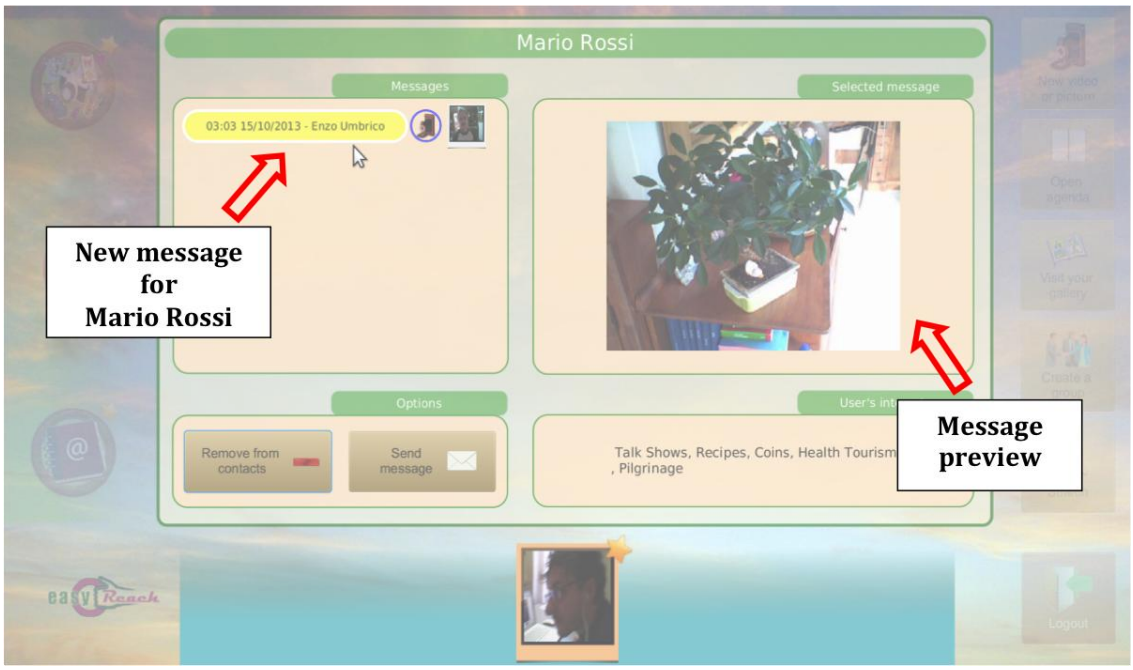

Figure 35: Updated Conversation Screen

Going back to our example, let us suppose that the user selects the image shown in Figure 34 and clicks on the “Send Message” button. The system will inform the remote server that a new message for Mario Rossi has been sent and the server will notify this event to the Mario Rossi’s STB-Client when the latter logs in. The sender user, after sending the message, will be automatically redirected on the Conversation Screen, updated with the last message sent, as shown Figure 35.

To conclude this part of the walkthrough, let us now imagine that the user (User1) is navigating the system and, at one time, another user (User2), independently if already included in the contact list or not, sends a message to User1. The system will notify the event by animating the *EasyReach Logo* in the bottom left part of the main screen (Figure 36).

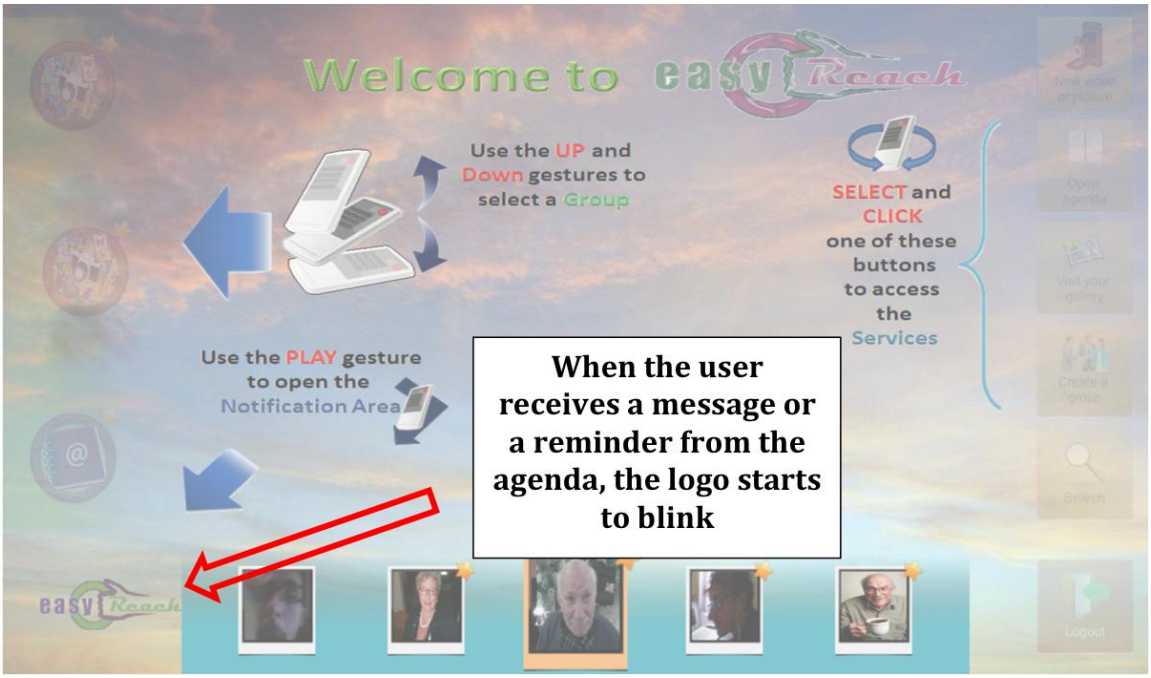

Figure 36: Animated EasyReach Logo

By using the *Play Gesture* with the Remote Control, the user can access the Notification screen (Figure 37): here the system shows all the incoming notifications (left box) about new messages sent by other users or posted to the groups the user is subscribed to. In the right box, instead, the system will show all the reminders sent by the Agenda Agent. As already said, in this particular example the user receives a message from another contact and therefore the system shows the message object related to this event, in the left box.

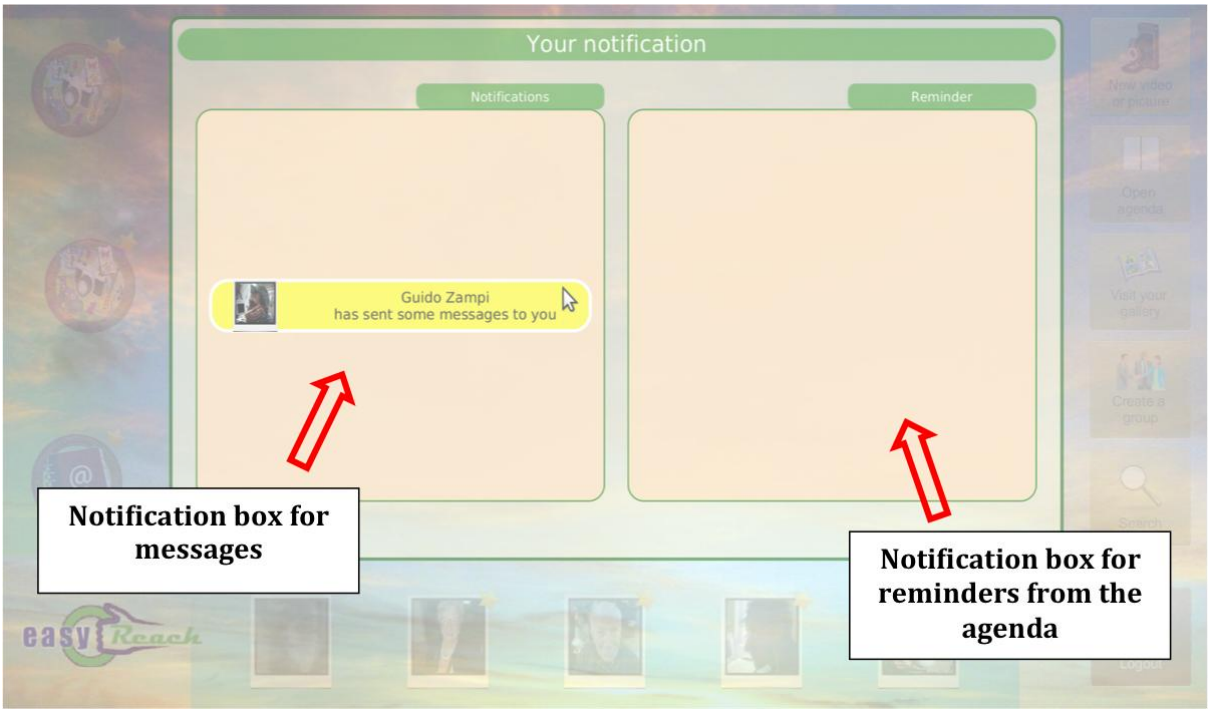

Figure 37: Notification Screen

By clicking on the message object, the system automatically redirects the user to the Conversation Screen, highlighting with a red colour all the unread messages, as shown in Figure 38.

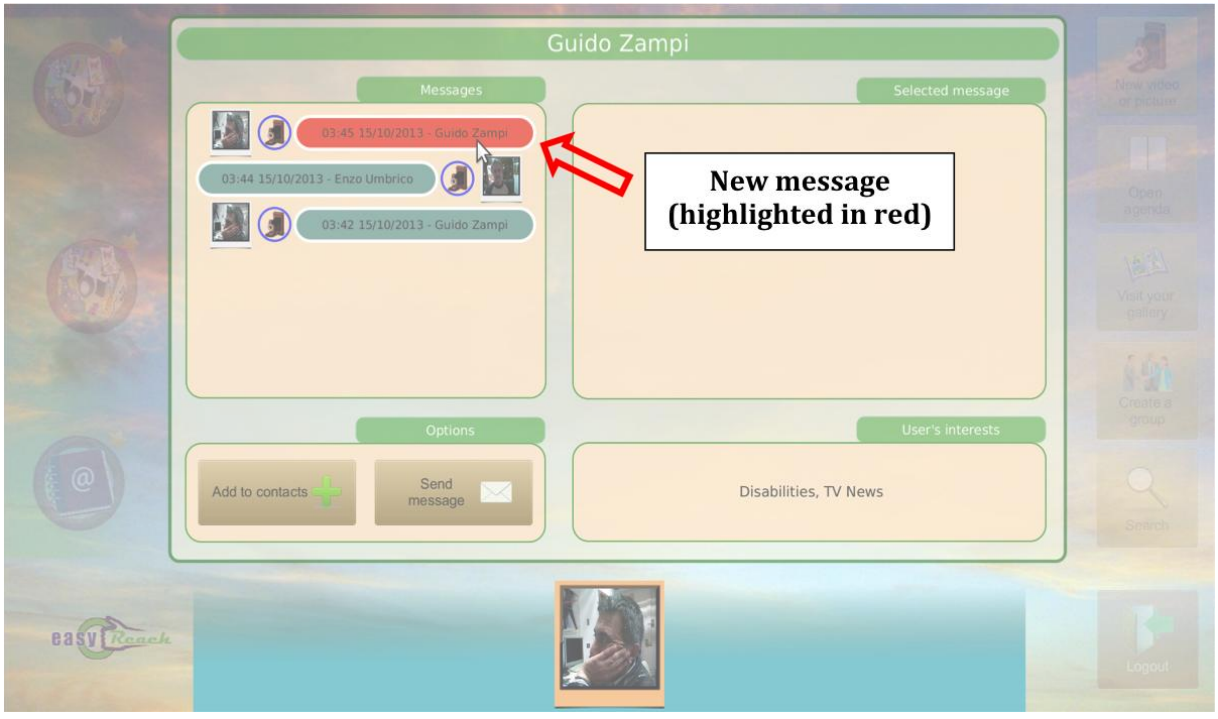

Figure 38: Conversation Screen and new Message

Figure 39 depicts an UML Interaction Diagram, which describes the interactions among elements composing the system back-end when a new notification is generated. When a user accesses the EasyReach system the *ModelController* initializes and starts the *NotificationListener* element as already described in Figure 11 of section 4.1.

The *NotificationListener* runs in background during user session and periodically checks for any unread messages or posts by calling the `checkNewMessages()` method of the *ElderActionService*, which in its turn calls the `getUnreadMessages()` of *PersistenceManagerService*. Note that in this case, unlike in other operations, no session is checked (and renewed) because this is a background activity automatically executed by the system at a cyclic rate. If one or more message is found, the *NotificationListener* notifies the *ModelController* by passing the list of new messages.

Subsequently, the *ModelController* generates the notification event and notifies the *SocialEnvironment*, which alerts the user by means of the blinking logo signal described previously. Then, when the user switches to the notification area, the *SocialEnvironment* updates the view and displays the list of the new messages received.

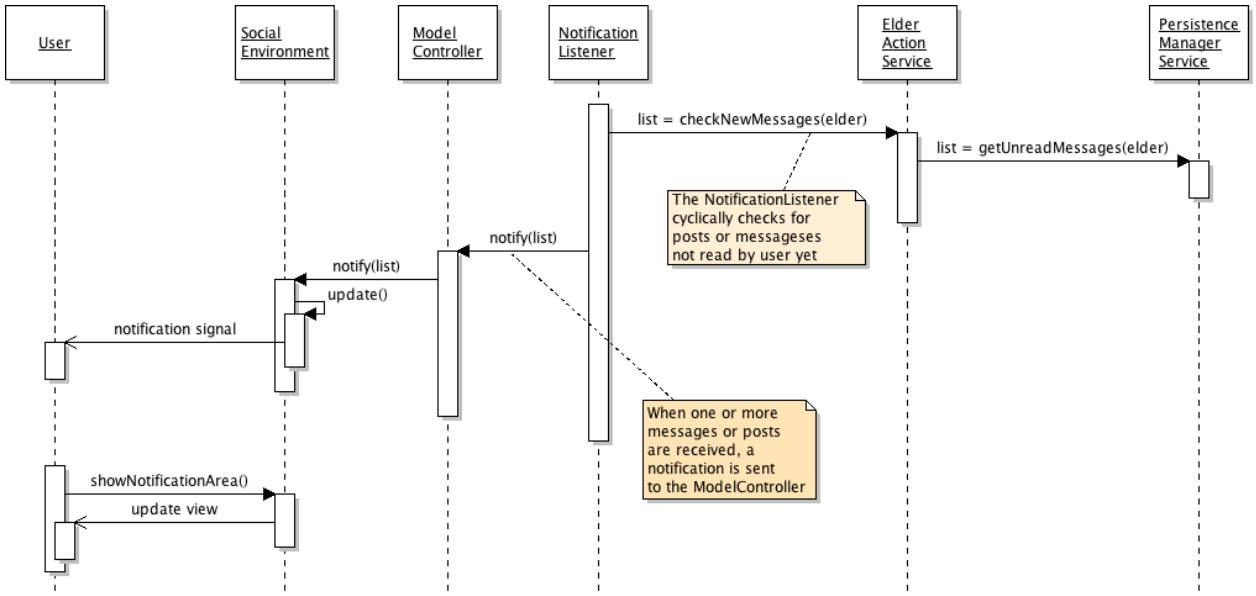

Figure 39: Receiving notifications

5.2. Creating groups of people that care for a certain topic

It might be often the case that the best way to discuss about specific topics is organizing dedicated and/or thematic groups. The Easyreach system allows the user to easily create and/or participate to discussion groups that are dedicated to specific themes of interest, themes that are clearly visible during the groups' browsing.

For instance, let us suppose that the user is interested in recipes; through a simple search, the user can immediately discover one or more groups whose discussions are centered upon cuisine issues. The user can therefore enter the chosen group, and immediately start reading/acquiring all previously posted messages therein - structured according to a bulletin-board organization and therefore readily available for reading, and possibly participate to the discussion, e.g., by proposing new recipes or asking further instructions about already present ones. The recipes can be completely described and explained by audio and/or video messages, while pictures describing the look of the cooked dishes can be posted on the group's message board.

Of course, the same features exemplified above can be exploited for any other case where the possibility of sharing multimedia content is essential to implement a truly meaningful and immediate social environment. For instance, in a thematic group about stamp collecting, it is certainly of great importance to share the pictures of the possessed stamps in order to favour comments, exchanges and comparisons; in a model building group, it might be helpful to have the possibility to view videos showing, for instance, part mounting or painting techniques shared by other users, etc.

In all the cases where a specific thematic group is not found by the user, or she/he wishes to create her/his own for any reason, this is certainly possible in a few "clicks". A typical case where this feature might be required is relatively to the generation of "family groups", i.e., groups composed solely of relatives and close friends. Indeed, not only should these groups be created on purpose by the elderly user, but they must be "private", i.e., accessed by a subset of contacts. By creating such a private group, the user could for instance share contents about her/his personal health status at all times to a sister or brother, also members of the same group, without incurring the risk that "uninvited" contacts might read the group's shared contents. The "contact search" feature offered by the system is particularly handy when creating private groups, as it is the only way to look for specific users whom the personal assistant did not automatically add to the contact list because of the scarce overlap in common interests; indeed, the composition of groups of relatives must be independent from the possible existence of shared interests among the participants.

In the following, we provide a walkthrough example relatively to the creation of a new group, focusing on the typical procedure the user use to create it. From the main screen (Figure 9), the user selects the "Create a group" option and the system will visualize the "Create new group" screen, as shown in Figure 40.

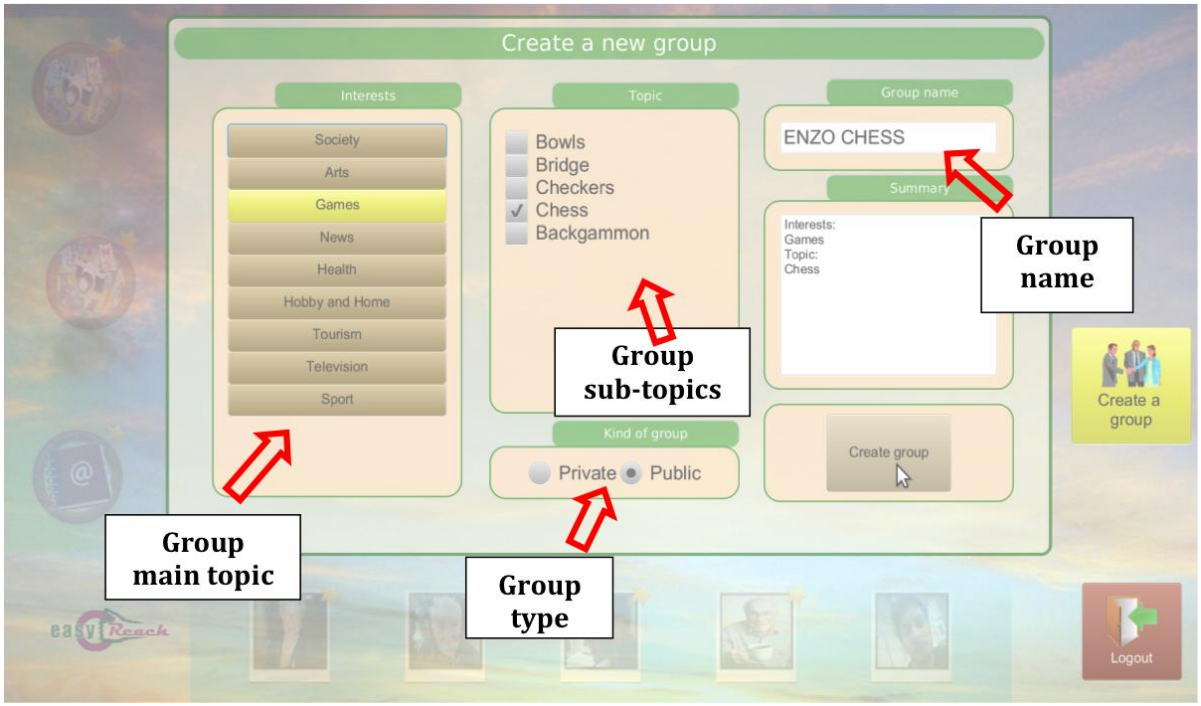

Figure 40: Creating a new group

From here, the user can choose the main topic of the group he/she intends to create from a list of "main topics", a set of sub-topics that depends on the previous selection, the type of group he/she is going to create (public or private) and the group's name. It should be underscored again at this point that the circumstances where the user is required to provide textual input have been minimized; however, in the rare cases this could not be avoided, this operation has been facilitated by means of a virtual keyboard (see Figure 31) that can be fully operated using the RC, as described in Section 5.1. Back to the example, the first choice the user has to do is the one related to the main topic of the group (Interest): in the example, the user selects the "Games" interest and a list of sub-topic is printed in the central box (the one related to the sub-topic).

In this particular case, the user selects the "Chess" sub-topic and decides that the group will have the name "Enzo Chess". At last, the user decides that the group will be a Public one (visible to everyone; instead, a Private group is visible only to the user's Personal contacts), and finally clicks on the button "Create Group" to finalize the group's creation. As a final result of this operation, the system will present the user with the newly created group page, as shown in figure 41.

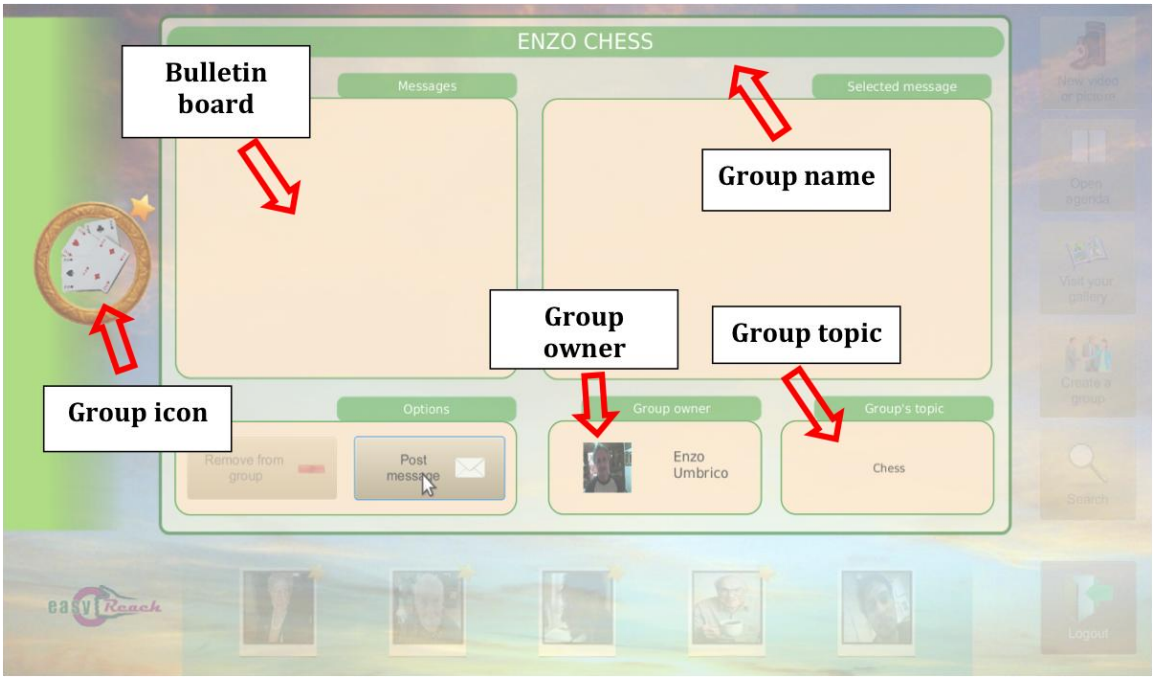

Figure 41: New group page

5.3. Organizing "interface" groups with existing organizations

For an elderly user constrained at home for a long period of time, it might be useful to keep in contact with the public organizations she/he used to attend, for instance in order not to lose track of the possible social initiatives that such organizations might coordinate. If, for example, the user is a frequenter of the local parish (or of a senior center, etc.), she/he might want to be kept updated of the possible trips and/or religious events that the parish is organizing in the future, in order to participate despite the fact that she/he is temporarily unable to physically join the party. On the parish's side, the EasyReach public client might be used to provide all the possible information on request, giving extra instructions when necessary, e.g., by publishing trip schedules, pictures and/or videos of the possible destinations, favouring the exchange of opinions among the possible participants, etc. Such information exchange can occur either through the established parish's EasyReach group (i.e., shared publicly), or through personal communication (e.g., sent directly to the user's single EasyReach account).

Independently of the current physical status of the users, the possibility to equip public organizations and/or services with an "EasyReach account" can be interesting in order to provide the elderly users with a "fast track" to use for requesting bureaucratic information and/or executing some of the operations that typically require long and tiring queues at the office desks. For example, the generic information about the management procedures of the post office accounts could be directly available on a dedicated EasyReach group. Not only could the users visit the group and acquire all necessary data from the group's bulletin board, and/or possibly ask for clarifications to the post office employee designated

to moderate the group, but they would have an easy way to share opinions, recommendations and experiences among one another directly, circumstance which might even be used to improve the organization's services by further tailoring such services to the specific needs of the elderly users.

Let us imagine for example that the user is forced at home for any reason (health problem, etc.), and despite this unlucky situation, he/she would like to keep in contact with its parish's group, and that he/she wishes to get updated about any events organized in the parish he frequents. The user, in the same way he can add a new contact to his/her contact list, can look for any group related to its parish (be it for instance, the Saint Amedeo's parish). So, the user can enter the Search Screen as explained in the previous sections, and directly search for the Saint Amedeo's Parish; the only difference with the previously described search procedure is that the user has to select, in this case, the "Group" button for the search, instead of the "User" button (recall the *Search* screen in Figure 30). The rest of the procedure, as well as the presentation of the possible results of the search, remains exactly the same.

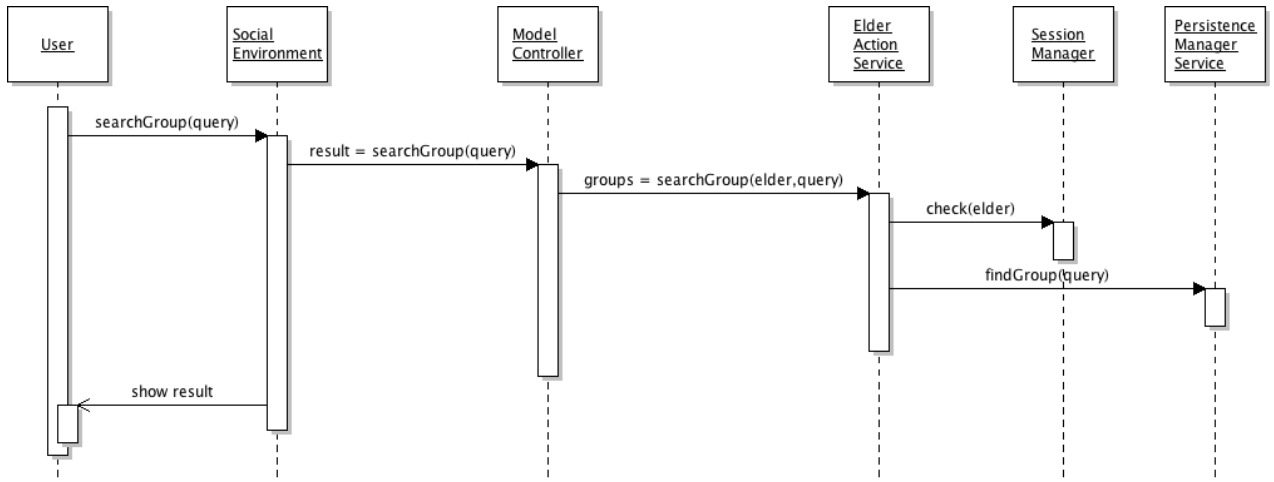

Figure 42: Search for a Group

Figure 42 shows an UML Interaction Diagram describing the interactions among system elements when the user makes a search request for groups. The procedure is very similar to the one already described in Figure 22 of Section 4.6, when searching for a new user. The *SocialEnvironment* element receives the request for searching a group and it calls the `searchGroup()` method of the *ModelController*. The ModelController sends the request to the *ElderActionService* by calling the `searchGroup()` and specifying the query and the user making the request. The *ElderActionService* first checks user session by means of *SessionManager* and, if the session is already valid, the *ElderActionService* calls the `findGroup()` method of the *PersistenceManagerService*, which executes the query on the system database and returns the groups matching the query. The result of the search group operation is depicted in Figure 43. Once the search results are shown, the

user can directly access the group's page by clicking on the selected item, and the system will immediately redirect the user to the selected Group Screen, as shown in Figure 44.

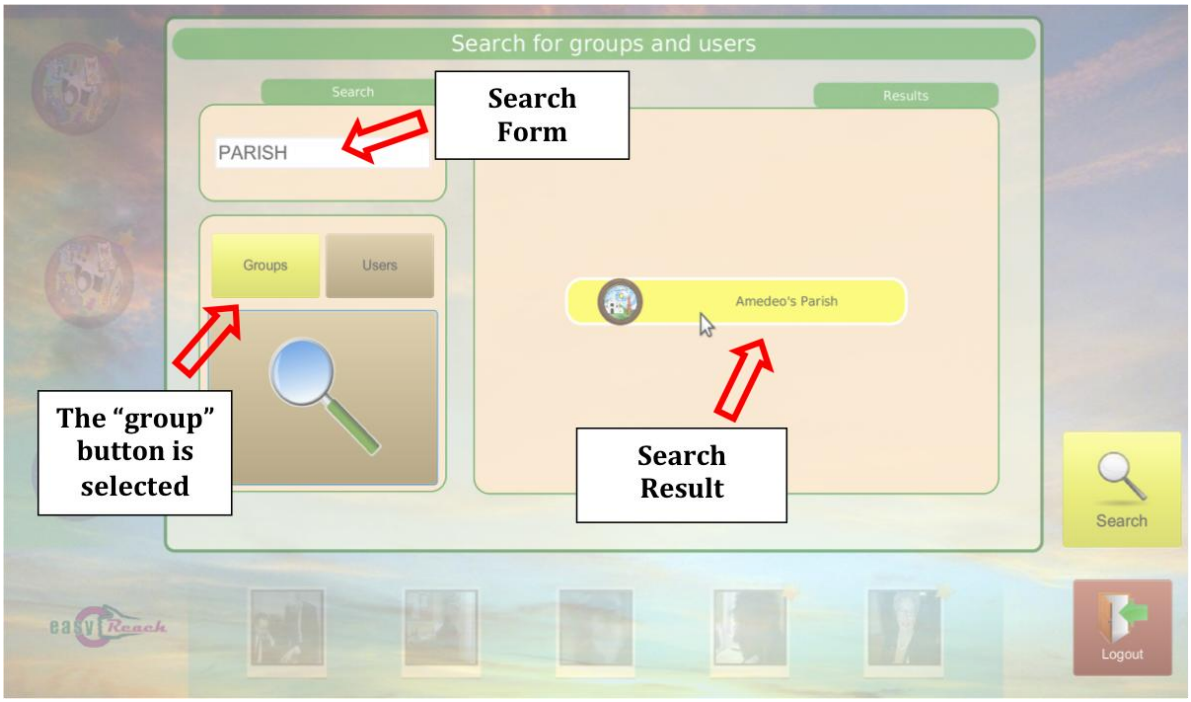

Figure 43: Search for a specific group

Here, the user can scroll the group's bulletin board, where all the messages placed by all the group's users are pinned. The system will utilize a *Lazy Loading* approach for the messages: the content of the messages will be downloaded only if the user selects it. In this way, the communication bandwidth is lightened and the loading time of the Group screen is shortened (the same technique is applied for the Conversation Group and the messages exchanged between two users).

In the particular case shown in Figure 44, the user selects a message to read: the system will show in the Message preview box the content of the selected message. Of course, the user can use also this page to post new messages in the group, exactly as he/she can do for any single contact.

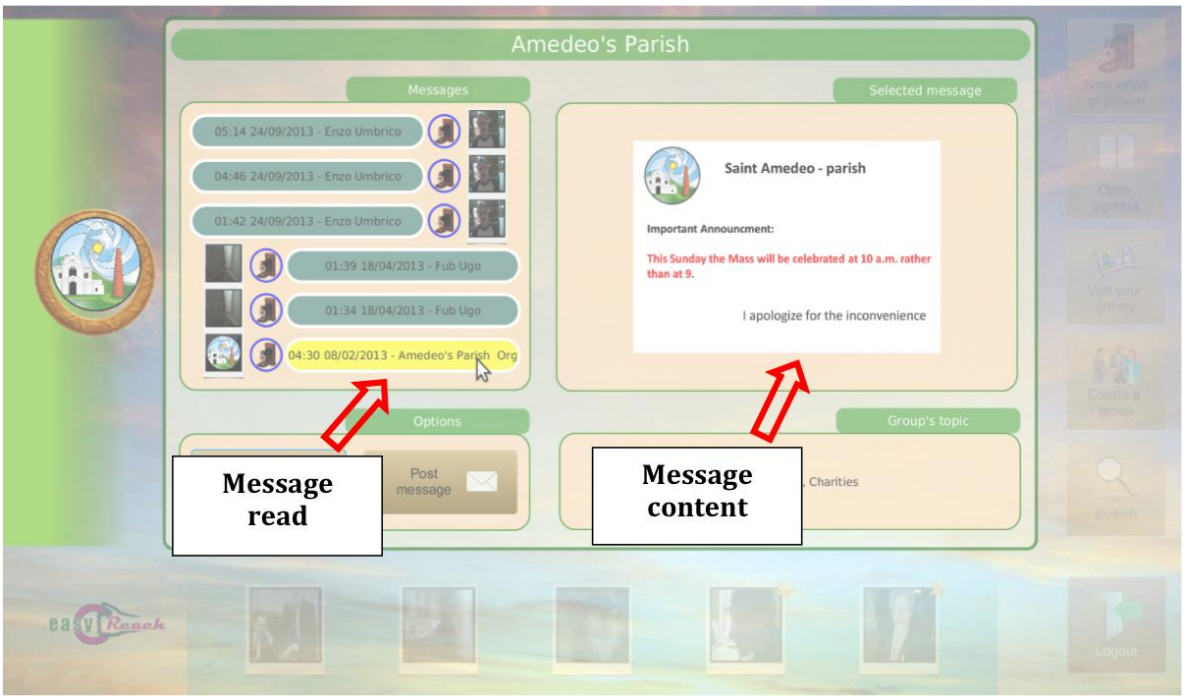

Figure 44: Example of group Interaction

5.4. Organizing help sessions where a skilled user can help or train other users

The aspect related to the sharing of one's experiences with other elderly users is very important in order to keep the feeling of being part of a community and avoid social isolation. Yet, in some cases, having the possibility to share with others the expertise acquired in a lifetime of professional activity can be of even higher importance. In fact, it is known that retirement can generally bring along some negative repercussions to the retired elderly, even depression. In those cases, giving the elderly people a way to keep themselves socially useful and or professionally active, may represent the key to their well-being.

In this respect, many examples can be presented where the EasyReach system can play an essential role. Let us imagine how many problems the elderly people must daily face relatively to the commonhold they live in. In such cases, a former condominium manager might provide all necessary help and/or information, by creating and managing an EasyReach group about commonhold management issues. Again, the EasyReach search procedure would make this task simple and enjoyable. Any user who wishes to have information from an expert person may simply access such group and look for the topic of interest or, if such topic is not found, directly ask specific questions to the group's creator. The added value is represented by the fact that not only would the users receive

information from a former expert, but they would have the possibility to share their own opinions and past experiences in a common forum.

Other typical topics of strong interest for the elderly are about the small repairs every house should regularly undergo. In such cases, the help provided by former plumbers, electricians, home appliance technicians, etc., even in terms of simple professional advice, could be greatly appreciated. Even more so, as all the multimedia information that can be exchanged in the EasyReach system easily lends itself to both explaining the issue at hand (e.g., the user can directly provide pictures of the problem she/he is suffering at home) and also to communicate its resolution (e.g., the expert user can share a video that explains the correct operation sequence to execute in order to solve the problem).

Lastly, another interesting possibility of application of the system is related to the organization of tutorial sessions (e.g., an EasyReach help group) where expert EasyReach users may provide useful information to the “newbies” to help them improving their knowledge and/or familiarity of the system. On the one hand, the elderly tutors would feel involved in an activity that fosters socialization; on the other hand, the “apprentices” would profit from the experiences of other users who very probably faced the same utilization problems and therefore can offer a greater understanding of the new user’s difficulties. In other words, learning the system “from within” the system, and stimulating a full immersion experience during the learning process, may reveal a further step towards a deeper system’s acceptance. In time, not only can the user take advantage of the EasyReach system to get help in any topic of any interest he likes but, on the other hand, he can create one or more help groups in order to share its expertise and lifetime knowledge.

In order to provide a walkthrough example, let us assume that the user would like to create a personal group about chess but he does not remember the correct procedure. In this case, the user can search for some help group about the EasyReach system (as explained in Section 5.3) and ask there for some help. Let us suppose that the search was successful and that the user has found (and accessed) an EasyReach help group in the system (see Figure 45). At this point, the user is presented with at least two choices; he/she can inspect the messages already present in the group to see whether the topic of interest has already been addressed and answered, or he/she can immediately prepare a message that explains his/her help request. The “Messages” box at the left of the screen shown in Figure 45 lists all the posts already exchanged in the group, both sent by the users and by the group’s moderator. After sending a video message where the user explains the problem, the moderator of the group eventually answers to him by posting, for example, a video tutorial in which she shows the steps for creating a new group. A preview of the video is immediately visible by simply selecting the response message, in order to facilitate the contents’ selection.

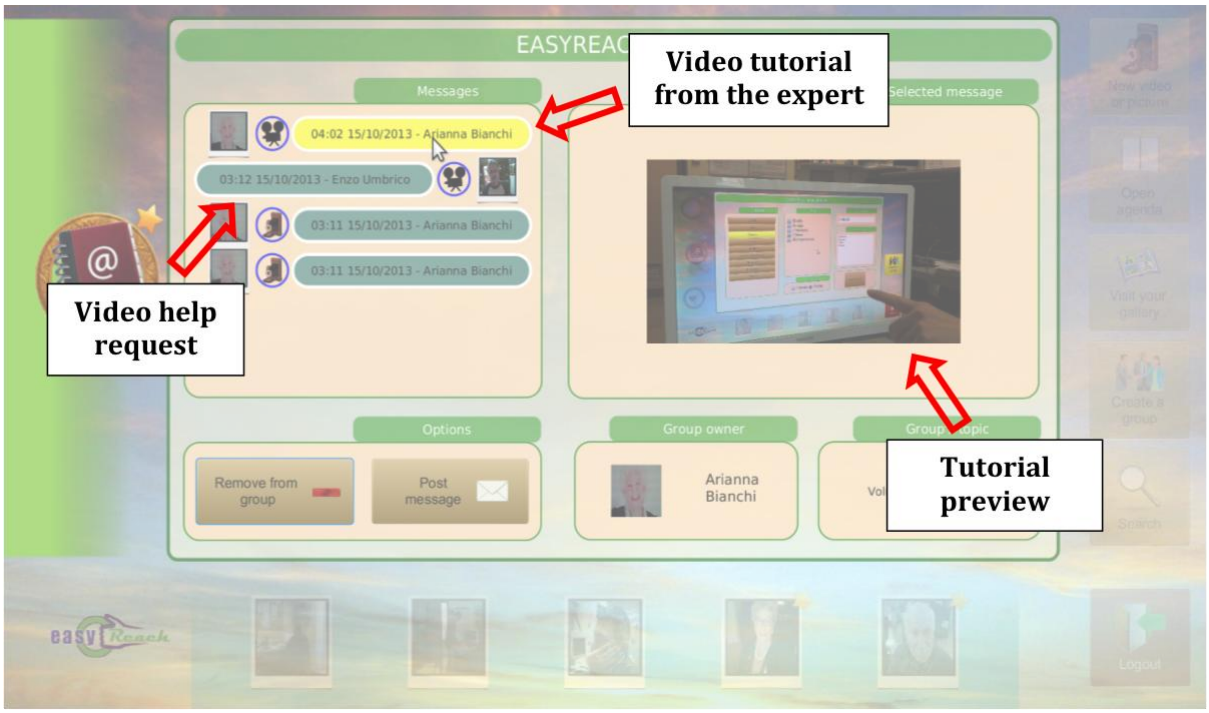

Figure 45: Example of help session

6. Conclusions

This document presents the final description of the design of the services that compose the EasyReach User Interaction Functionalities implementing the social network-like services, target of the project. In the design and in the implementation of such functionalities, great importance has been given to the necessity to tailor both the system utilization procedures and the offered services to the specific requirements of an elderly users' basin, starting from the perceived difficulties on behalf of the pre-digital divide population in approaching last generation web technology.

In terms of the selection of the hardware devices to be used to interact with the system (whose motivations have been already explained in previous deliverables), the realization of the current system version has remained coherent with respect to the original idea proposed at the beginning of the EasyReach initiative, though the user-centered design approach used during the whole development of the system, has at times highlighted the need to apply a number of "in-process" slight modifications w.r.t. what initially designed.

The same applies relatively to the software side, with the only exception of the selection of the exploited network technology, where the choice finally fell on the Java EE Web Service technology, given its total openness, flexibility, and versatility with respect to the EasyReach needs. Given the high importance for the EasyReach system of the technology selection that ensures all the network communication capabilities and services, it has been essential to select a technology that most naturally merged with the software

solutions previously adopted throughout the system (e.g., for the graphical user interface), thus guaranteeing both software homogeneity and easiness of implementation.

On the base of the description of the overall system architecture and of the existing relationships among all the architectural software modules, as well as of the system's hardware components, the organization of the services offered by the system are provided in this document. Such services are presented in a bottom-up fashion: firstly, the set of low level elementary activities is described, underscoring the main guideline of "utilization immediacy" and proactiveness that inspired their design and implementation; secondly, a set of exemplary social services is described, each built on top of the previous base activities, and therefore maintaining the characteristic of simplicity of use which has been constant objectives within EasyReach.

The fulfilment of the project's objectives and the overall appreciation of the system on behalf of its intended basin of users has been tested through a set of pilots with real users (see [1,2]), with encouraging results. The robustness of the system, as well as its easiness of use have also been largely verified: in this respect, the EasyReach system has proved to be (i) easy to transport and set-up, and (ii) sufficiently robust to be deployed and used in different conditions and settings.

Figure 46: The EasyReach System shown at the AAL Forum in Norrkoping, Sweden

It is worth mentioning the interest aroused by the EasyReach Project also in the scientific area of Intelligent Applications; we have presented the stable architecture of the EasyReach system in a publication at the IEA/AIE 2013 International Conference (see [5]), and recently we have been notified of the selection in the short list of the best papers for the special issue of the Springer journal "Applied Intelligence" (special issue on Advances in Applied Artificial Intelligence). We are currently working on an extended version of the paper, where both the technical and the social aspects of the EasyReach initiative will be thoroughly presented.

In conclusion, Figure 46 shows a moment during the recent system demonstration which took place at the Ambient Assisted Living Forum 2013 in Norrkoping, Sweden. In the highly specialized assembly of the AAL JP, the idea of providing elderly people with social functionalities by means of a non-invasive solution through the deployment of equipment that did not require any complicated setup or a re-engineering of the homely environment has been much appreciated by both the stakeholders and the stand visitors. In particular, the idea of using tools the elderly are already familiar with (i.e., the TV, the Remote Control, etc.) to control the system and the interaction with the social channel has been commented very positively; in addition, several entities have demonstrated interest for the possible customization of the technology within different application settings.

References

[1] "*Report of the pilot results*", EasyReach Deliverable D6.2

[2] "Socio-economic assessment of the developed solutions and set of guidelines for further development", EasyReach Deliverable D6.3

[3] "*Complete Agent Implementation and Documentation*", EasyReach Deliverable D3.3

[4] "*Prototype of the smart Remote Control*", EasyReach Deliverable D4.1

[5] R. Bisiani, D. Merico, S. Pinardi, M. Dominoni, A. Cesta, A. Orlandini, R. Rasconi, M. Suriano, A. Umbrico, O. Sabuncu, T. Schaub, D. DSAloisi, R. Nicolussi, F. Papa, V. Bouglas, G. Giakas, T. Kavatzikidis and S. Bonfiglio. "*Fostering Social Interaction of Home-Bound Elderly People: The EasyReach System*" In M. Ali, T. Bosse, K. V. Hindriks, M. Hoogendoorn, C. M. Jonker and J. Treur (Eds.): IEA/AIE 2013, 26th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE'2013), Lecture Notes in Artificial Intelligence (LNAI), Vol. 7906, pp. 191-201, 2013;