socialize
service oriented assisted living infrastructure

WP 2– Development of SOCIALIZE system components

Document: D2.5 Socialize Forum Service

Date: <15/11/2016>

Confidentiality: ☐ Public ☑ Restricted ☐

Version: ☐ Draft ☐ Final ☑

PROGRAMME

DOCUMENT INFORMATION

Acronym of lead partner for the deliverable	ISTI-CNR
Work package	WP 2 - Development of SOCIALIZE system components
Contractual date of delivery	September 2016
Date of delivery	November 2016
Nature	Report
Dissemination level	Public

Document Responsible	ISTI–CNR (IT)
Author(s)	Dario Russo, Vittorio Miori

Release	1
Version	1
Date	15/11/2016

Revision History

Date	Version	Author	Paragraph added/modified	Description
15/11/2016	1.0	ISTI-CNR		Final version

Table of Contents

1 Executive Summary

The primary objective of this service is to promote the discussion and exchange of opinions conducted online, about issues of concern to the elderly population.

The goal has been achieved through an innovative solution designed and built for the purpose. It was crated a web platform for socialization directed to the discussion of issues and problems related to the world of the elderly. Although there are already numerous platforms of Social Networks and Forums, it was decided, after considering the results emerged from the deliverables of WP 1, to create a new application in order to introduce functionality otherwise difficult and poorly integrated within this service.

The most important feature lies in make emerge the contents posted in the platform by elderly that they found interesting and useful. This is realized through a mechanism similar to the "like" mechanism. In this manner it is encouraged and improved the possibility for the elderly to socialize and to have evidence of the most popular and significant discussions, thus avoiding the dispersion in those unfrequented or with little relevant contents. With this similar "like" mechanism, seniors can also actively contribute even to discussions with a simple "click" without be forced to enter contents. The categorization of contents is expressed in areas and sub-areas.

The most important characteristics of this service are:

- the simple and intuitive use;
- at most two levels of complexity of nesting of discussions in threads;
- decision-making mechanism with evidence of the most current and participated discussions;
- the participatory mechanism to resolve disputes and controversies (typical of the elderly population) to reach to a common decision;

to facilitate participation by offering the opportunity to actively contribute to decisions, even without being forced to post written interventions just with a "click".

2 INTRODUCTION

Socialize Forum is a web service that enables elderly to actively participate and to question for issues of public interest related seniors. To this end, the web interface and the application functioning is appositely designed to be used by elderly and in particular for those are hostile to the use of technology.

The implementation phase involved the creation of a web portal that allows older people to actively participate in the issues concerning their daily lives, through interfaces and standard tools, so as to allow access to as many people as possible. In order to achieve this, particular attention was paid to the ease of access to the system.

After evaluating the requirements and objectives to be achieved, a set of open-source applications were considered to be used for the implementation phase.

So, to implement this service, most used solutions to build web application have been taken into account. In particular where considered the best known *CMS* (*Content Management System*) as *Typo3*[1], *Plone*[2] and *Drupal*[3], and the more recent *Symfony*[4] framework.

Because of the constraints dictated by the user requirements, and the lack of flexibility and ease of use of *CMS*, this service was implemented using *Symfony*.

Symfony is a framework based on *php5*[5], one of the best known and adopted languages to make dynamic web applications. The most important features of this framework are:

- easy to define templates and then the user interface;
- a cache system to speed things up on the server;
- the separation of application logic and display of the content;
- the support to *Ajax*[6].

With Symfony is used *Apache*[7], one of the most famous multiplatform and reliable open-source web server.

1 https://typo3.org

2 https://plone.org

3 https://www.drupal.org

4 http://symfony.com

5 http://php.net

6 https://en.wikipedia.org/wiki/Ajax_%28programming%29

7 http://www.apache.org

3 The implementation

The system implementation provides for four types of user: *anonymous*, *registered*, *moderator* and *administrator*.

3.1 Users

An *anonymous* user is a person who has not logged into the system by entering his access and therefore he is not. He has free read access to the system content, but he cannot express opinions or leave traces in some way.

A *registered* user is a subscribed user and he is acknowledged by the *Socialize* platform. He can read the contents of the system and can also express his ideas:

- creating and / or declaring interest in a question proposed by other users.
- creating and / or giving approval to a response.
- inserting comments, replies to comments, attachments and links.

He can also:

- report any content that does not comply with the policy;
- report users as spammers;
- add elderly to his list of friends;
- write to other users.

The *moderator* user has the same rights and features of the *registered* user. In addition, he is responsible for the content posted by elderly and of their behaviour. He can change the focus of a discussion, insert, modify and delete news and write to users.

He is able to temporarily or permanently obscure irrelevant contents (that can be reported by users of the system), and temporarily or permanently block access to a user if his behaviour is doesn't respect the policy. He can insert, edit and delete news and events, write to all users interested in a question.

The *administrator* user has the privileges to administer the system changing and customizing capabilities, contents and settings. In particular, he is able to enable/disable moderator users.

3.2 Functionalities

The system implements four functionality groups:

3.2.1 Functionalities related to content

Threads will be arranged in a pattern that provides two distinct levels: the first one relates to the formulation of a question, the second one is the answer to that question. This will be designed to facilitate and organize discussions with a model that has only two levels of complexity. Indeed, any additional comments to the answers will be optional.

There will be also a voting mechanism (or liking) both of the questions and the answers, which allows bringing immediately to the user's attention the most attended threads (and therefore more current and participated). This will allow also to bring out shared decisions, following a democratically agreement.

Four system capabilities groups will be provided:

Capabilities related to the contents that allow facilitating discussions and allowing the emersion of shared decisions. Users who will be enrolled into the system will be able to propose questions and to discuss solutions or to insert comments.

The discussions between the users will be organized using the classical schema:

- emersion of discussions, users indicate questions on a particular topic. It represents a problem or a request that an user wants to submit to the community. It must belong to a thematic and a sub-thematic area;
- answers for each question, users can specify the proposed solutions. They are the contributions of the other users of the community that express opinion and answers.
- liking of answers, ability for users to express and withdraw their approval to the proposals contained in the answers. Users can give or revoke their vote of interest or favour to a question or an answer contributing to bring out the most important contents;
- comments to the answers, any question or proposed solution can be commented by users with opinions or suggestions.

Registered users will can insert their own content such as questions, answers, comments, replies to comments, attachments and web references; they will can give likings and approvals to the content posted by other users in order to obtain a "score".

Users will can indicate territorial references (geo-reference of the subject in a map) related to both questions and answers or comments. These references will be pinpointed and displayed in special maps within the sections of the forum.

There will be an *internal* messaging system however, it is possible to configure the system so that messages are forwarded to the email address indicated by the user if he makes this wish.

Furthermore, there will be an automated messaging mechanisms related to content modification.

There will be other capabilities only managed by moderator users such as: news, calendar, FAQ, contacts, credits and links

These functionalities enable to insert, and to manage the discussions inside the platform.

It is possible to geo-reference the subject of the question in a map.

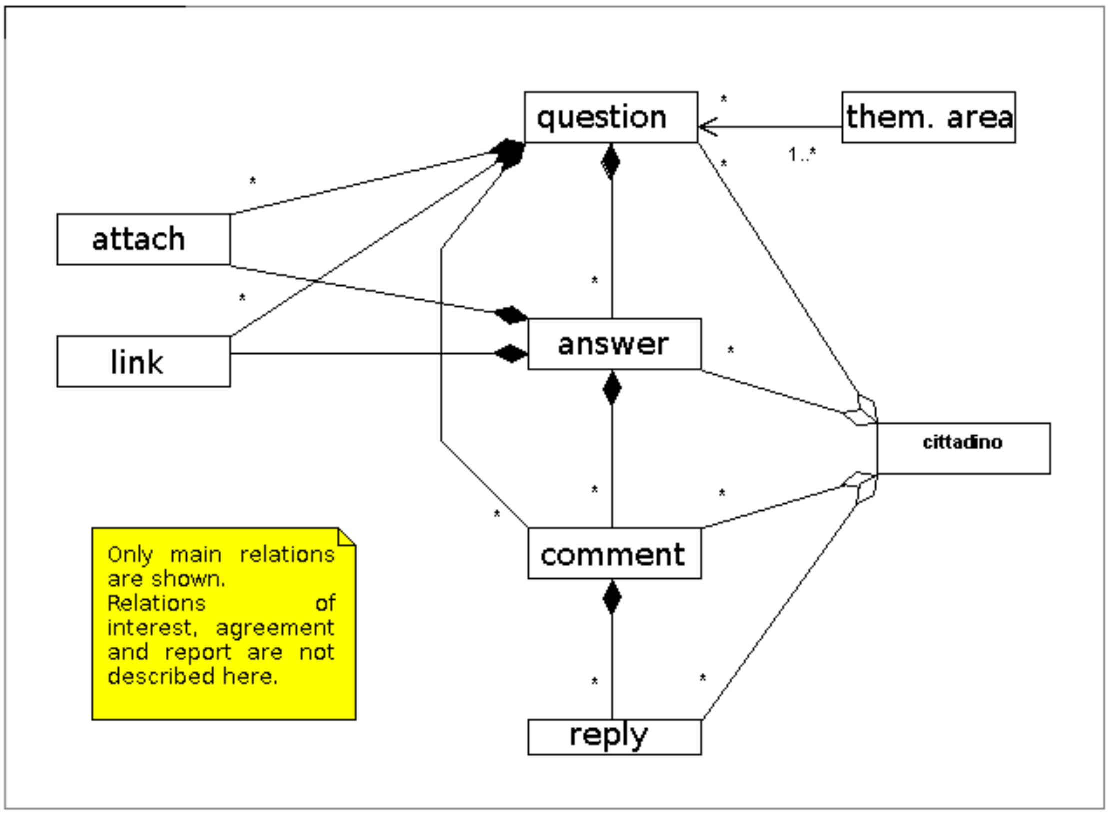
question
them. area
1..*
attach
link
answer
cittadino
comment
reply
Only main relations are shown. Relations of interest, agreement and report are not described here.

Fig. 1 Lifecycle of a discussion

A question (Figure 1) is created by a *registered* user and it is associated at least to a specific topic. A question can contain attachments and links to external resources (attach, link). Other citizens can express their interest to the problem and leave a comment to the problem.

An answer is referred to a question and is also made by a *registered* user, it is a suggestion for a solution of a problem. The suggestion is presented by a user, who becomes the owner of it. Other users can declare their acceptance to the answer and express any comments, participating in the discussion. Comments can be followed by replies (reply) that are not subject of expression of relevance.

An answer may be modified over time, but only by his manager (owner user or a moderator). The change of an answer involves a notification to all those involved in the discussion.

According to amendments, users can withdraw their liking (or possibly give it, if they had given it).

To insert a content, the user has to fill a form composed by fields to insert a title, a brief description and a possible address to see the geo-reference (if useful) of the question on the map. The description is a text describing the problem.

Each question belongs to one topic and sub-topic

It is possible to insert your own contents such as problems, proposals, comments, responses to comments, attachments and web references; to express approval with respect to the contents published by other users in order to obtain a sort of "score", to report content considered censurable, explaining the reasons.

Moderator user can temporarily or permanently obscure contents and state the reasons for that. This operation makes the selected contents no longer visible except in designated areas.

3.2.2 FUNCTIONALITIES RELATED TO MESSAGING

The messaging system is internal, meaning that messages are stored in the database and are not sent by default via standard e-mail protocols. However, it is possible to configure the system so that messages are forwarded to the user's e-mail address if he o wishes to do so. This choice is personal and each user can do it by editing their own profile from their personal page. So, by default, massages are sent via standard protocols *SMTP*.

Received messages are displayed in the inbox list when they arrive. Messages can be moved to the archive list or can be removed.

The messaging system is used principally by the system to send notification to *registered* and *moderator* users, and by elderly to communicate privately and directly among them.

Each user can add or remove users to their friends list, which is used exclusively to reach users' personal pages.

The system messages principally include alerts to events related users' contents. In particular, the system alerts:

- the user when his question has received an answer;
- the user when his question / answer has received a new comment;

- the user when his question / answer received an interesting / favour vote;
- the user (with email) when he has been blocked;
- the user when his content has been obscured;
- moderators when a content has been reported;
- moderators when a content has been overshadowed by another moderator;
- moderators when a user has been flagged as a spammer;
- moderators when a user has been blocked;

3.2.3 CMS FUNCTIONALITIES

These CMS functionalities are managed and accessible exclusively by the *moderator* users:

- edit and remove contents directly from the page that show them.
- post news, which will be published in the specific section, sorted by descending order by date;
- enter a new event which will be published in the specific section, sorted by descending order by date;
- FAQ: list of questions and answers;
- Contacts: contact list to request information;
- credits: list of authors and maintainers of the site;
  link: Opportunities for external resources list

3.2.4 FUNCTIONALITIES RELATED TO USERS

The most important user-related functionality is authentication to the system. Authentication is used to recognize the user user of the system, and their credentials are verified so that the available functions can be limited according to the type of user. User authentication is done according to the standard username and password mechanism. Through identification it is possible to determine the typology of user and the operation that he is authorized to perform.

A user who is not registered to the system but has the required personal data can register by entering the personal data necessary for the correct use of the system.

A *registered* user may report the behaviour of another *registered* user if he considers it to be incorrect.

A *moderator* may temporarily or permanently deny access to the system to a registered user if his or her behaviour is found to be irrelevant or inappropriate, giving reasons for this.

Each user has a personal page where are summarized his activities and where he manages the list of his friends, which is used principally to reach the users' personal pages.

Automatic messaging mechanisms linked to the change in content status are also foreseen. In particular, the system warns:

- the user that one of his problems has received a proposal;.;
- the user that one of his problem / proposal has received a new comment;
- the user that he has received an approval for a problem/proposal;
- favourable users (including the owner user) that a proposal they have voted has changed status;
- the user (with email) who has been locked;
- the user whose content has been obscured;
- moderators that a content has been reported;
- moderators that one content has been obscured by another moderator;
- moderators that a user has been reported as spammer;
- moderators that a user has been locked;

4 DEVELOPMENT

The system's development activities have been articulated taking into account the criticality related to the timing of actions and dedicated resources.

The coordination of development activities was supported by some team collaboration and source code management tools. In particular, the working group has adopted an open source version control system, called "subversion", with a repository for the application's source code.

4.1 PRELIMINARY ANALYSIS

In accordance with the project schedule, a preliminary phase was launched and preliminary studies were carried out. These studies focused mainly on the approach for structuring the final system.

In this phase, two possible options were considered:

- Customization of a Content Management System: this solution would have been very quick, since it would have allowed to start with a base already working, but not without substantial intervention complexity. The systems considered were primarily *Typo3* and *Plone* (the first in *php4* and the second in *python*). Both presented a very valid approach from the point of view of content management, but with regard to the aims of the SOCIALIZE project, they would have required a strong personalization, with results that were not sufficiently satisfactory. In this sense, the group considered the possibility of possibly releasing a plug-in module for such *CMS*, in *Open Source* format. This would have been in line with the "community" philosophy of such software.
- Use of a Web Framework: certainly more challenging and ambitious solution, but with a reasonable certainty both to obtain a result that would achieve the goal and to create a platform of greater technical importance. In this respect, several innovative frameworks and programming languages were identified. *MVC* systems were mainly considered, i. e. with a strong conceptual and operational separation between application logic and graphical presentation.

  The final choice was to "*Symfony*" an *Open Source* product in *php5*. The group analysed the potential and highlighted the critical points, considering them to be essentially controllable or eliminable in the final system. Although functional to operational decisions, this initial analysis phase provided a basis for the subsequent code evaluation and final testing activities

At this phase some basic choices were made:

- The choice of a rather common programming language and not covered by commercial licences. This allows anyone to develop, collaborate and modify the software using a popular, well-known, documented programming language without having to purchase licenses for use, and therefore without increasing development costs.

- The choice for the use of a database technology that was also easily available and documented. The database is one of the crucial points in developing a web application. It has been possible to have a reliable, stable and widely documented open-source database and thus fully in line with reusability. The database is an archive organised in such a way as to allow the management of data such as entry, search, deletion and updating by software entities. The choice was made on *My-Sql*. *My-Sql* is an open-source database particularly appreciated in web oriented applications for its speed and reliability. The *Symfony* framework natively supports the database by simplifying and improving application development code.
- The possibility of using existing database technologies in the application context. If the application context already supports the chosen database, application development is greatly simplified, ensuring a higher degree of security and completeness.
- The web server. The web server is the program that is responsible for providing on request a browser, a web page. The choice was to use *Apache*. *Apache* is a reliable and widely used open-source web server. It can manage and support all the features offered by *php* and *Symfony.*

  Symfony is a framework for *php5*, one of the most popular and adopted languages to make web applications dynamic. The most important features of this framework are the simplicity in defining templates and therefore the user interface, a cache system to speed up operations on the server, the separation between application logic and visualization of contents and support for *Ajax*.
- The use of standard development models. The programming techniques and the standard development process allow a correct progress process, controlled and easily controllable by anyone, in every phase, even with the help of standard tools.
- Code structuring in the direction of a good separation between functionality, graphic presentation and general settings. The common model commonly used to implement the separation between functionality, presentation and graphics is MVC (Model - View - Controller). Through this model it is possible to carry out any possible interventions on one of these elements, making the other two as little as possible, creating the least possible dependence on each other.

During this preliminary phase, the working group was engaged in a series of coordination and brainstorming activities, preceded by periods of testing and evaluation of the various solutions chosen. The total duration of this phase was about 4 months.

4.2 *Operational Scheduling*

The development work was organized by separating a number of activities that were informally associated with workpackage and assigned to the work units within the technical group.

The following subactivities were identified:

- Database design and relational data model
- Implementation of the data model within the framework and implementation of business logic
- Graphic layout development and integration
- Development and integration of the geographical information access module and map display
- Testing
- Deploying the software

With regard to the database design, a relational scheme has been produced on which the data model was built.

This model was then put into practice during the software implementation activity. This phase involved the CNR operating unit for a further 3 months.

In parallel, the definition of the graphic aspect of the product was started. The integration of graphic layouts into the source code in the form of operating templates (html) was particularly challenging.

The two previous activities were then merged into a first beta version, placed on a CNR server, in order to start the first testing activities. Subsequently, agreements were made to plan a minimum level of intervention on code and testing by the whole group.

The integration of georeferenced maps into the final software was still developed in parallel. A collateral part of the testing was aimed at evaluations on Deploy software. This was done in a particular way, on the performance level of the final product, its stability and security, in order to provide important feedback for the final refining.

5 MVC Model

The reference model used to customize the system is the *Model-View-Controller* (*MVC*). The model is based on the separation of tasks between software components that play three main roles:

- the model contains the data and provides methods for accessing it;
- the view displays the data contained in the model;
- the controller receives user commands (usually through the *view*) and implements them by changing the status of the other two components.

With this model, the platform has been designed with particular attention to achieve a good level of separation between the fundamental elements that make it up:

- the operational part of the data processing;
- the graphic part, showing the results;
- the logical part, describing the data and their structure.

During development, the logical part was further modulated by grouping the functionalities that produce dependency so you can add, remove and modify modules without having major repercussions on the rest of the application, ensuring a high reusability factor and easy customization.

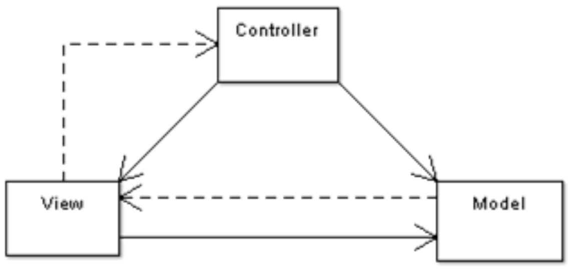

Fig. 2 Model-View-Controller paradigm

By appropriately modifying one or more of the three components making up the model, different levels of customization can be obtained:

1. Unmodified *Model* and *Controller*. Changing *View* Only.

   This is the most common case that will arise. The application is satisfactory in all its parts: functionality and type of data. Only the part of the graphic presentation is changed, as is natural. In this case, you need to act on the "graphical templates".

2. *Controller* not modified. Changing *View* and *Model*.

   The features are confirmed but we want to apply them to different types of data with different graphics.

   This is the most complex case of reuse (apart from complete redesign). In this case, the platform is to be applied to a context for which it was not originally conceived. For example, to apply the software in a context where users will be elderly with a different cognitive degree.

   In this case we follow the same type of approach that Socialize evaluated at the beginning (deciding then on the most challenging road of an ex-novo design) that is to use existing content management systems and modify them so that they can adapt to our needs.

3. *Model* not modified. Changing the *View* and *Controller*.

   In this case, the platform is applied to a similar context to the initial one. However, their behaviour and business logic are changed. It is a more advanced reuse of simple graphic customization. In this case we modify the source code allowing a complete customization of the platform, even outside the context originally planned

4. Unmodified *Model*, *View* and *Controller*.

   Although this may seem like a banal case of reuse, it was actually foreseen during the design phase (particularly because that the platform can be applied to different service providers, with a single graphic layout). In this case (figure 4), configuration mechanisms have been set up in such a way as to allow the fundamental parameters (name of the organisation, logo, characteristics of the organization, etc.) to be changed through "*yml*" configuration files (a format used to serialize data that can be easily understood and modified by humans) and the "*microcontent*" (the set of basic messages provided by the web platform (e. g. in emails confirming the registration or in descriptive snippets).

Therefore, a variety of possible interventions with increasing complexity are configured, which find in the following figure a possible schematisation

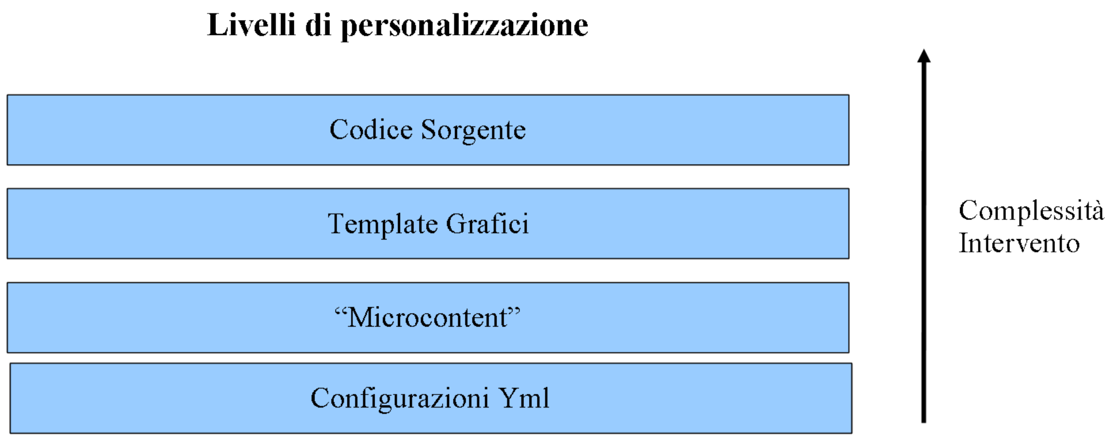

Fig. 3 Reuse Levels