Grant Agreement no. 215056

Title:	***D6.2b - Community application prototype***
Editor:	*Elsa Prieto, Atos Origin, Spain (ATOS)*
Reviewers:	*Bernd Ueberschaer, Leibniz Institute of Marine Sciences. Germany (IFM-GEOMAR)*
	Katrin Kristjansdottir, Center for Usability Research & Engineering, Austria (CURE)
Identifier:	*D.6.2b*
Type:	*Deliverable*
Version:	*1.0*
Date:	*27/10/2010*
Status:	*Final*
Class:	*Public*

Summary

This deliverable D6.2b presents the first version of the PICOS community application prototype for the gaming community (gamer v1). It describes the new functionality, developments and modifications applied to the second prototype for the angler community (angler v2.5), which was the firm basis for the gamer mobile application. These new features are derived from the requirements collected from the gamer community and are described in the R2 investigation report (WP4). Some angler requirements that were postponed in previous implementation phases due to time constraints are also addresses in the gamers prototype. The priority for the implementation of feature follows that of the requirements in the internal document R2, this is "necessary requirements", "recommended requirements" and "helpful requirements".

The resulting prototype will be treated from WP3 (assurance), WP7 (user trials), WP8 (evaluation) and WP9 (dissemination).

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Members of the PICOS consortium:

Johann Wolfgang Goethe-Universität (Coordinator)	Germany
Hewlett-Packard Laboratories Bristol	United Kingdom
Hewlett-Packard Centre de Competence France	France
Universidad de Málaga	Spain
Center for Usability Research & Engineering	Austria
Katholieke Universiteit Leuven	Belgium
IT-Objects GmbH.	Germany
Atos Origin	Spain
Deutsche Telekom AG	Germany
Leibniz Institute of Marine Sciences	Germany
Masaryk University	Czech Republic

The PICOS Deliverable Series

These documents are all available from the project website located at http://picos-project.eu.

D2.1 Taxonomy	July 2008
D2.2 Categorisation of Communities	July 2008
D2.3 Contextual Framework	November 2008
D2.4 Requirements	November 2008
D3.1.1 Trust and Privacy Assurance for the Platform Design	April 2009
D3.2.1 Trust and Privacy Assurance of the Platform Prototype	November 2009
D3.3.1 Trust and Privacy Assurance of the Community Prototype	January 2010
D3.4.1 A summary of PICOS WP3 sub-phase 3.1 deliverables	August 2010
D4.1 Platform Architecture and Design v1	March 2009
D4.2 Platform Architecture and Design 2	September 2010
D5.1 Platform description document v1	October 2009
D5.2a Platform prototype 2a	May 2010
D6.1 Community Application Prototype 1	December 2009
D6.2a Community application prototype 2	April 2010
D7.1a Trial Design Document	December 2009
D7.1b Trial plan for the second community prototype	September 2010

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.
The PICOS project receives research funding from the Community's Seventh Framework Programme.

D7.2a First Community Prototype: Lab and Field Test Report	February 2010
D7.2b First Community Prototype: Field Trial Report	August 2010
D8.1 Legal, economic and technical evaluation of the first platform and community prototype	April 2010
D9.1 Web Presence	February 2008
D9.2.1 Exploitation Planning	April 2009
D9.2.2 Exploitation Plan 2	March 2010
D9.3.1 Dissemination Planning	April 2009
D9.3.2 Dissemination Report V2	March 2010

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

The PICOS Deliverable Series

Vision and Objectives of PICOS

With the emergence of services for professional and private online collaboration via the Internet, many European citizens spend work and leisure time in online communities. Users consciously leave private information; they may also leave personalized traces they are unaware of. The objective of the project is to advance the state of the art in technologies that provide privacy-enhanced identity and trust management features within complex community-supporting services that are built on Next Generation Networks and delivered by multiple communication service providers. The approach taken by the project is to research, develop, build trial and evaluate an open, privacy-respecting, trust-enabling platform that supports the provision of community services by mobile communication service providers.

The following PICOS materials are available from the project website http://www.picos-project.eu.

Planned PICOS documentation

- Slide presentations, press releases, and further public documents that outline the project objectives, approach, and expected results;
- PICOS global work plan providing an excerpt of the contract with the European Commission.

PICOS results

- *PICOS Foundation* for the technical work in PICOS is built by the categorization of communities, a common taxonomy, requirements, and a contextual framework for the PICOS platform research and development;
- *PICOS Platform Architecture and Design* provides the basis of the PICOS identity management platform;
- *PICOS Platform Prototype* demonstrates the provision of state-of-the-art privacy and trust technology to leisure and business communities;
- *Community Application Prototype* is built and used to validate the concepts of the platform architecture and design and their acceptability by covering scenarios of private and professional communities;
- *PICOS Trials* validate the acceptability of the PICOS concepts and approach chosen from the end-user point of view;
- *PICOS Evaluations* assess the prototypes from a technical, legal and social-economic perspective and result in conclusions and policy recommendations;
- *PICOS-related scientific publications* produced within the scope of the project.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Foreword

Deliverable D6.2b is a collective work by the WP6 development team, whose members are listed below. We would like to thank those members of PICOS consortium who contributed directly or indirectly to the production and refinement of this document. Special mention goes to Marek Kumpost (Masaryk University, Czech Republic) for acting as a link with the gamers community and helped with the definition of certain features.

Since this is the last official deliverable related to WP6, we would also like to express our gratitude to past members of PICOS who contributed to both earlier deliverables and prototypes. Special acknowledge to Bernd Ueberschaer from Leibniz Institute of Marine Sciences, former contributor to WP6 and reviewer of the present document.

The WP6 Team

ATOS, CURE, GUF, HPF, ITO, and TMO.

Reviewers:

Bernd Ueberschaer, Leibniz Institute of Marine Sciences. Germany (IFM-GEOMAR)

Katrin Kristjansdottir, Center for Usability Research & Engineering, Austria (CURE)

Editors:

Elsa Prieto, Atos Origin, Spain (ATOS)

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Table of Contents

1 Introduction 11

1.1 Scope and Context of D6.2b in PICOS lifecycle. *11*
1.2 Deliverable Structure *12*

2 The gamer application from the user perspective 13

2.1 Relationship between Anglers and Gamers prototypes. *13*
2.2 Description of features. *14*
2.2.1 Gamers profile manager 14
2.2.2 Presence and availability management 16
2.2.3 Contacts management 19
2.2.4 Policy Manager 19
2.2.5 Privacy Advisor 20
2.2.6 Public Community 21
2.2.7 Public Repository 24
2.2.8 My Files 27
2.2.9 Sub-Communities 29
2.2.10 Location Based services 31
2.2.11 Real-time Content Sharing: Communication and Notification Means 37
2.2.12 Notifications 39
2.2.13 Advertising services 39
2.2.14 Revocation 40

2.3 Usability improvements *41*

3 High-Level Design of Gamers Client Architecture 42

3.1 Scope and purpose *42*
3.2 Application Architecture Overview *42*
3.2.2 General Functions 44
3.2.3 Non-Functional Aspects 44
3.2.4 Development Tools and Libraries 44
3.2.5 Development Environment 44
3.2.6 Deployment 44

4 Gamer application prototype Design 45

4.1 Gamers profile manager *45*
4.2 Presence and availability management *45*
4.3 Contacts management *46*
4.4 Policy Manager *50*

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

4.5 Privacy Advisor *52*
4.6 Public Community *52*
4.6.1 Access restriction to published content in the Public Community 52
4.6.2 Content access history 57
4.6.3 Subscription to receiving new available content in threads of the public community..... 58
4.7 Public Repository *60*
4.7.1 Access restriction to published content in the Public Repository (Categories) 60
4.7.2 Access Restriction to Category Files 62
4.8 My Files *63*
4.9 Sub-Communities *65*
4.9.1 Access restriction to published content in Sub-Communities 65
4.9.2 Access Restrictions based on date conditions 68
4.10 Location Based services *70*
4.10.1Point Of Interest 70
4.10.2Point Of Interest Recommendation 72
4.10.3Private Site 73
4.10.4List Nearby Users..... 74
4.11 Real-time Content Sharing: Communication and Notification Means *75*
4.12 Notifications *75*
4.13 Advertising services *76*
4.14 Revocation *77*

Appendix A List of R2 implemented requirements in Gamers prototype. 78

Bibliography & References 79

List of Tables

Table 1: Summary of new features in Gamers prototype 14

List of Figures

Figure 1. WP6 framework and dependencies 11
Figure 2: Gamers profile 16
Figure 3: Variable number of hobbies in the profile 16
Figure 4: User Status Management: creation of a status condition. 17
Figure 5: User Status Management: view status condition. 18
Figure 6: Display a contact's calendar. 18
Figure 7: Share contact list screen. 19
Figure 8: Privacy Manager screens 20

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Figure 9: Public Community main screen 21
Figure 10: Rule creation attached to content in the public community screen. 22
Figure 11: Edition of content rules in the public community. 22
Figure 12: Access history in public community. 23
Figure 13: Notification for new available content in the public community. 23
Figure 14: Rule creation attached to content in the public repository 24
Figure 15: Display of rules attached to a category in the public repository. 25
Figure 16: Deletion of a rule attached to a category in the public repository. 25
Figure 17: Modification of a rule attached to a category of the public repository. 26
Figure 18: Access history in public repository 26
Figure 19: Subscribe to notification for new available content in the public repository. 27
Figure 20: My Files screen and example of file creation. 28
Figure 21: File transfer from My Files to Public Community 28
Figure 22: Rule creation attached to content in the sub-community screen 29
Figure 23: Content access history in Sub-communities. 30
Figure 24: Access history in Sub-communities 30
Figure 25: Creation of a Private Site. 31
Figure 26: List Private Sites. 32
Figure 27: Definition of rules attached to Private Sites 33
Figure 28: Creation of a new Point of Interest. 34
Figure 29: "List Nearby Users" link in "Community" tab. 34
Figure 30: "Nearby Users" screen. 35
Figure 31: "Nearby Users"options. 35
Figure 32: Send message to nearby user. 36
Figure 33: Create contacts to nearby users. 36
Figure 34: Start chat with nearby users. 37
Figure 35: Show nearby users on map. 37
Figure 36: Real-time Content Sharing: file in chats. 38
Figure 37: Real-time Content Sharing: archive chat 38
Figure 38. Commercial Point of Interest screen 40
Figure 39. Revocation from PICOS community 41
Figure 40: Overall Architectural Overview 42
Figure 41: Orchestration layer in Gamers prototype 43
Figure 42: Calendar Class Diagram 45
Figure 43: Use cases of sharing Contact-List 46
Figure 44: Sequence diagram “show contacts of contact” 48
Figure 45: Class diagram “sharing contact list” 49
Figure 46: Sequence diagram “create contact list rule” 50
Figure 47: Class diagram “Privacy Manager” 52
Figure 48: Access restriction to thread content (Posts) class diagram 53
Figure 49: Access restriction to thread content (Posts) sequence diagram 54
Figure 50: Display, modify and delete thread rules class diagram 55
Figure 51: Display, modify and delete thread rules sequence diagram 56
Figure 52: Content access history class diagram 57
Figure 53: content access history sequence diagram. 58
Figure 54: Notification of new available content in forum thread class diagram. 58

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community’s Seventh Framework Programme.

Figure 55: Notification of new available content in forum thread sequence diagram. 59
Figure 56: Access restriction to content in the Public Repository (Categories) class diagram. 60
Figure 57: Access restriction to content in the Public Repository (Categories) sequence diagram. 61
Figure 58: Access restriction to content in the Public Repository class diagram. 62
Figure 59: Access restriction to content in the Public Repository sequence diagram. 63
Figure 60: Shared Desk class diagram 64
Figure 61: Create new shared Desk sequence diagram 65
Figure 62: Invite contact to a shared Desk sequence diagram 65
Figure 63: Class diagram for access restriction to published content in the Sub-Communities. 66
Figure 64: Rules attached to published content Sequence Diagram 67
Figure 65: Class diagram for access restriction to based on date conditions in the Sub-Communities. 68
Figure 66: Rules attached to Sub-Community content Sequence Diagram 69
Figure 67: POI deployment diagram. 70
Figure 68: POI usecase diagram. 71
Figure 69: POI recommendation activity diagram. 73
Figure 70: Private Site usecase diagram. 73
Figure 71: List Nearby Users sequence diagram. 75
Figure 72: Offline notifications diagram 76
Figure 73: Advertising Services usecase diagram. 77

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

List of acronyms

Abbr	*Abbreviation*
API	*Application Programming Interface*
GPS	*Global Positioning System*
GUI	*Graphical User Interface*
HLD	*High Level Design*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
ICT	*Information and Communication Technology*
ID	*Identity (also Identifier)*
IdM	*Identity Management (also Identity Manager)*
IM	*Instant Message*
J2ME	*Java 2 Mobile Edition*
LBS	*Location-Based Services*
LWUIT	*Lightweight User Interface Toolkit*
MVC	*Model-View-Controller*
N/A	*Not applicable*
OTA	*Over the Air*
PA	*Privacy Advisor*
PICOS	*Privacy and Identity Management for COmmunity Services*
PID	*Partial ID*
POI	*Point of Interest*
RPC	*Remote Procedure Call*
SC	*Sub-Community*
UTC	*Universal Time, Coordinated*
WP	*Workpackage*

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.
The PICOS project receives research funding from the Community's Seventh Framework Programme.

1 Introduction

The present deliverable D6.2b is the first and only prototype deliverable for the new community of online Gamers, closing the implementation activities for PICOS project on the client side (WP6). This new prototype aims at showing that PICOS solution and concepts can be extended and adapted to the other groups' needs building on a pre-existing architecture common to all of them.

1.1 Scope and Context of D6.2b in PICOS lifecycle.

D6.2b is the result of the work performed by WP6 development team from March to September 2010, in line with the planned activities for the second cycle [1] and in synchronization with WP5 work (platform side).

The main outcome of this activity is a new prototype for the second PICOS community, targeting the gaming community that will be tested in the scheduled WP7 trials (October 2010). This prototype will also go under an assessment and evaluation process by WP3 and WP8 teams and serve for dissemination purposes (WP9).

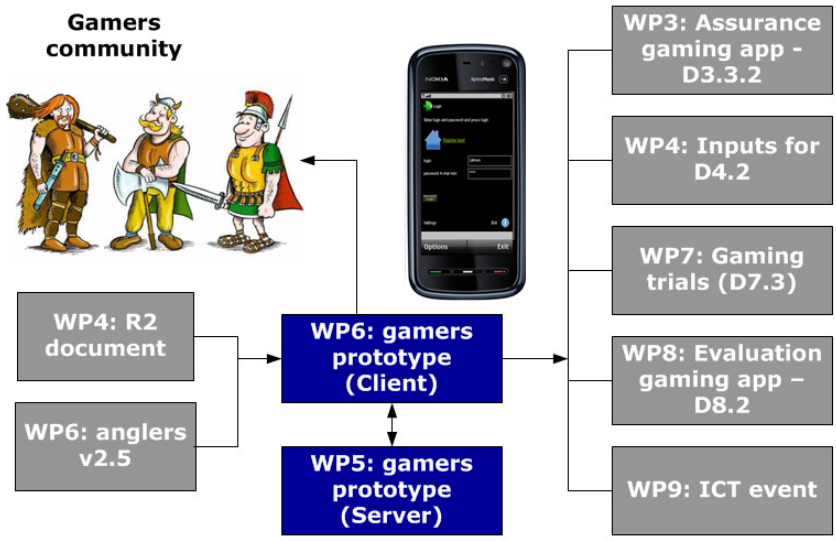


Figure 1. WP6 framework and dependencies

The starting point for this development phase was the investigation of requirements collected from the Gamer community in the internal document R2 [2]. R2 (WP4) contained new features to develop for the second prototype, but it also inherited some anglers requirements that had not been taken into account for the anglers second prototype. The list included a prioritized evaluation, based on the gamers' interests and needs, ranging from "necessary requirements" to "recommended requirements" and finally to "helpful requirements" that guided the work under WP5-WP6 team. The categorization was necessary because of the limited time resources of the development teams. Moreover, during the specification and implementation phases there was a direct link to the gamer community through the

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Masaryk University that was essential for clarifying the gamers requirements (R2 requirements) related to the implementation of the new features.

The degree of completeness of the final application was subject to several challenges:

- A tight planning for the development of the gamers prototype (including specification, design, implementation and testing phases) during the third year of the project. The WP interdependencies and the shortage of time couldn't give much leeway for any contingency in the prototype development to provide inputs to other WPs in due course.
- Large amount of requirements (29 requirements in R2), high complexity of new functionalities and adaptation of prototype for the new community.
- Fine tuning between WP5 and WP6. Some decoupling issues arose over the implementation period that had an impact on both sides of server and client that needed to be overcome.
- Other commitments: contributions to D4.2 (WP4) [7], technical support to anglers [8] and gamers trials [9] (WP7) and technical support to the ICT event team (WP9) [10].

To tackle these challenges, WP5-WP6 team agreed to follow three iteration phases according to the R2 priority, putting more emphasis on indispensable requirements and less on those who are just "nice to have". Appendix A provides a list of those requirements that were finally included in the application. This list shows that all necessary requirements and recommended requirements were covered; helpful requirements were covered to 80%, which represents a satisfactory figure.

1.2 Deliverable Structure.

This document is divided into four major parts:

- The first part (this section 1.2) is an introduction to the scope and context description of the deliverable D6.2b and the rationale of its structure and presentation of results.
- The second part (chapter 2) is an overview of the implemented functionalities from the user perspective.
- The third part (chapter 3) presents the high level architectural description from the client perspective. This section can be considered as an update of the architecture described in D6.1 chapter 2, with the changes introduced by the gamer implementation.
- Finally chapter 4 provides the technical description of the main features of Gamers prototypes (class and sequence diagrams).The structure of this chapter follows the use cases appearing in previous deliverables (D6.1 Community Application Prototype [3] and D6.2a Community application prototype 2 [4]) to ease the comparison between them.

We strongly recommend consulting deliverables D6.1 and D6.2b for supplementary information on the client application(s), as well as WP5 deliverables (D5.1[5], D5.2a[6] and D5.2b) for better comprehension of the complete prototype(s). Furthermore deliverable D4.2 [7] provides a rationale of the requirements here addressed.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

2 The gamer application from the user perspective

2.1 Relationship between Anglers and Gamers prototypes.

The Gamers application (D6.2b) is built on the Anglers application version number 2.5. (D6.2a[4]). Therefore the Gamers application has the same functionalities as the previous prototype. Nevertheless appropriate modifications and extensions have been made to adapt the anglers application to the specific needs of the gamers. For example, Angler-Specific Services (species summary functionalities), catch reports creation and Location Based Services (LBS) related to watercourses and fishing-spots were removed in the new application. Moreover new features have been incorporated according to the requirements compiled in the internal deliverable R2 [2]. These features are largely characterized by the implementation of new policy rules letting the user (the gamer) handle them, for instance rules applied to content management (threads, files) for a specific set of contacts during a valid period of time.

Another aspect that distinguishes the gamer from the angler solution is the absence of a Web Frontend (Anglersbase [11]) extending the mobile application to a PC environment. This extension was not part of Gamers requirements. Therefore gamers functionality is only available on the mobile application.

The main differences between anglers and gamers applications are summarized in the following table, along with a reference to the related PICOS concepts they relate to.

A short description of each feature can be found in section 2.2.

Application Component	Functionality	Nature	Related PICOS principle
Profile manager	Gamers profile	Extension	Privacy
Presence manager	Enriched status	Extension	Trust
Contacts management	Contact list sharing	Extension	Trust
Policy Manager	New version.	Improvement	N/A
Privacy Advisor	Enhance Content Awareness.	Extension	Privacy
Public community	Restriction to published content (contact condition)	New	Privacy
	Restriction to published content (date condition)	New	Privacy
	Content access history	New	Privacy
	Notification of new content available in public-community	New	N/A

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Public repository	Restriction to published content (contact condition)	New	Privacy
	Restriction to published content (date condition)	New	Privacy
	Content access history	New	Privacy
	Notification of new content available in public repository	New	N/A
My Files **(private room)**	Shared-desk	Extension	Privacy
Sub-communities	Restriction to published content (contact condition)	New	Privacy
	Restriction to published content (date condition)	New	Privacy
	Content access history	New	Privacy
	Notification of new content available in sub-communities	New	N/A
LBS	Social presence awareness (automated blurring)	Extension	Privacy
	Points of interests (POIs)	New	Privacy, Trust
	Information about nearby gamers	New	Privacy, Trust
Advertising services	Creation of advertisements based on commercial POIs (cPOIs) Subscription to advertising services.	New	N/A
Content Sharing	Real chats with files (images)	Extension	N/A
	Archive chats	Extension	N/A
Notifications	Offline notifications	Improvement	N/A
Membership	Revocation	Improvement	Trust

Table 1: Summary of new features in Gamers prototype

2.2 Description of features.

2.2.1 Gamers profile manager

In the second PICOS community, the member's profile was subjected to some modifications to append new attributes or adapt the existing ones in anglers application. The resulting root ID consists of the following attributes (grouped by pull-down sections). Blue color highlights what was added or modified in the profile in relation to anglers profile:

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Pseudonym (partial ID1)

Avatar image (Button "Add Image")

Main Info

- Family Name
- Age
- Gender

Hometown

- Country
- Locality ("City")
- Street Name
- Zip Code

Messaging

- Email
- Skype-ID (old "Instant message comm.")
- Phone Number
- Facebook (IM address 2)
- Internet Access
- Preferred contacts means

Hobbies info

- Hobbies (merged with Interests)
- Favourite Games
- Favourite Location
- Alliance Memberships (old "fishing clubs")
- Playing Style (old "fishing style")
- Travian epochs
- Blank attribute

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

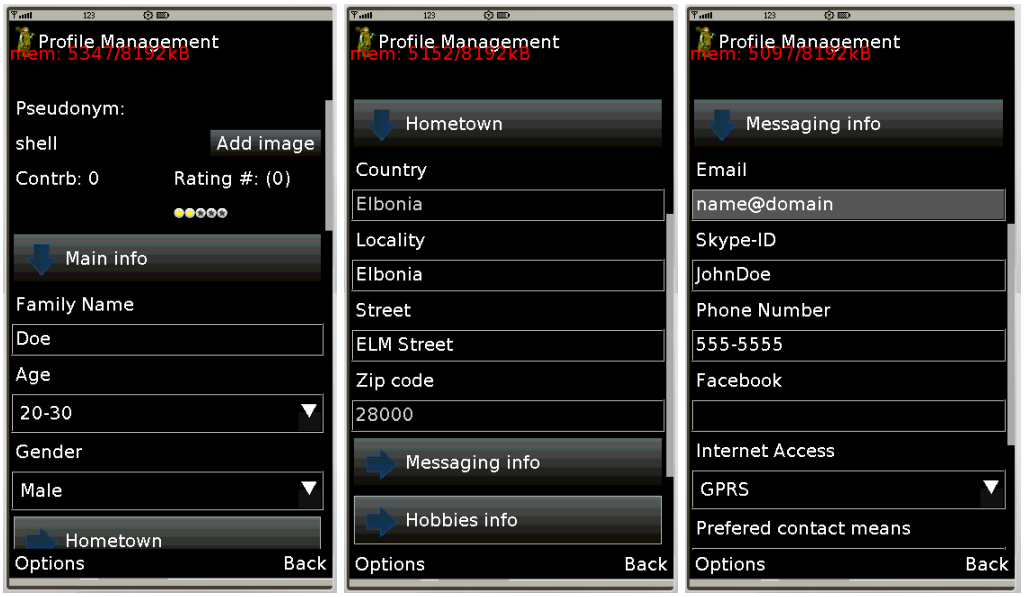


Figure 2: Gamers profile

When creating a Partial ID (PID) the following attributes cannot be changed: “Family Name”, “Age”,” Gender” “Street Name”, “Locality” “Country”, and” Zip code”.

One of the major changes in the profile is the option to store hobbies in different fields, using the corresponding buttons “Add hobby” or “Remove hobby”.

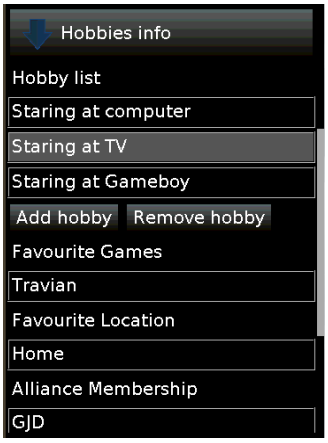


Figure 3: Variable number of hobbies in the profile

2.2.2 Presence and availability management

In contrast to the anglers prototype where the user could only set their status either as “Online”, “Offline”, or “Discreet”, the gamers application provides a way for the user to set availability status and see the availability of their contacts in a given date marked on a calendar, in order to organize meetings or strategies.

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community’s Seventh Framework Programme.

This feature is accessible from the navigation bar through two tabs:

- "Calendar" tab: the user will be able to display a calendar showing his own statuses (set previously), and he can also set new ones, entering dates (from calendar) and times.
- "Contacts" tab: the user will be able to select one of his contacts and press "show user calendar", which means the contact's calendar is displayed showing his set availability status

The following figures show how a user sets a new status for a specific period of time parting from the tab "Calendar" in the navigation bar.

After status configuration (availability status or another kind of status), the user's contacts will be able to see the user's calendar, as long as he authorized his contacts to see his presence.

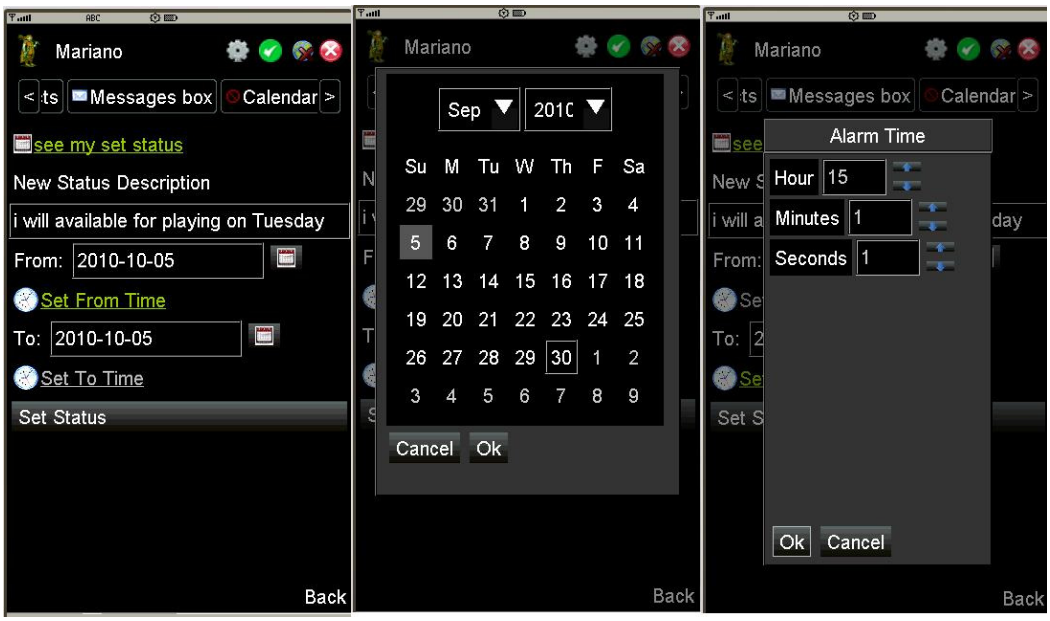


Figure 4: User Status Management: creation of a status condition.

Subsequently the user will be able to navigate his calendar to see those dates marked with a status condition. The highlighted dates indicate the existence of a status condition.

Clicking on them a summary of the status condition is displayed.

The link "Remove this status" permits the user to cancel the created condition.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

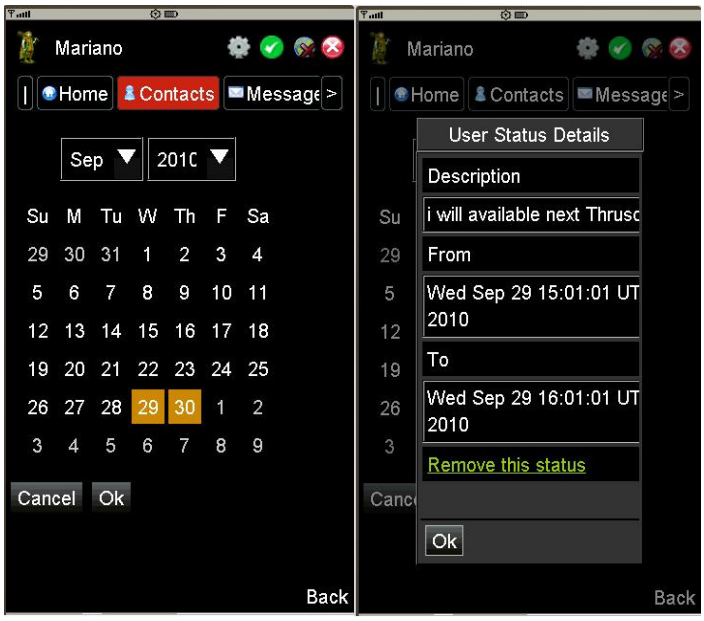


Figure 5: User Status Management: view status condition.

Moreover, a gamer could wish to see one of his contact's calendar in order to arrange a meeting, for example a date to play. In that case the user would go through his contact list screen, select one of them and press "Show User Calendar". The contact's calendar is displayed and similarly highlighted dates will indicate that the contact introduced a status condition. Clicking on these dates the condition is displayed. Unlike the example above, the user cannot edit his contact calendar, thus the link "Remove this status" does not appear.

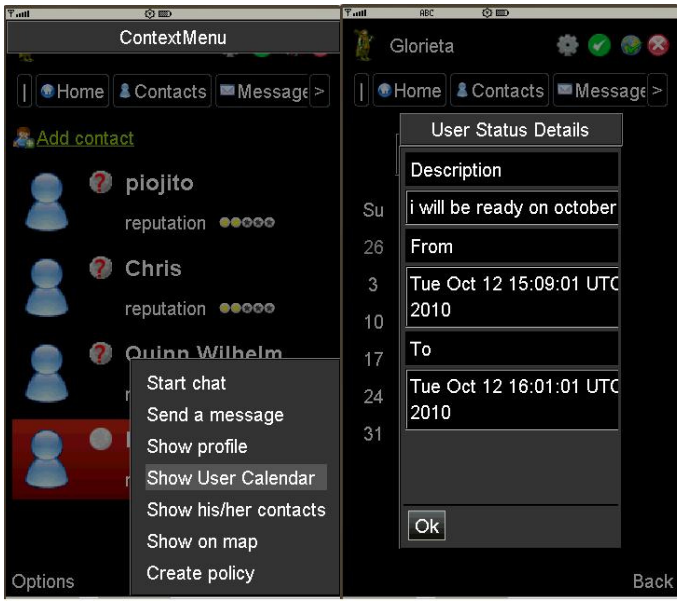


Figure 6: Display a contact's calendar.

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

2.2.3 Contacts management

2.2.3.1 Contact list sharing

As an extension of the contact management functionality, a user can share his contact list with a set of members of the public community or with a set of members belonging to a sub-community (SC). By using a context menu in the contact list ("press and hold" on a contact), these authorized individuals will be able to see the contacts ("Show his/her contacts"), the profile and also add them to their own contact list if the invitation is accepted.

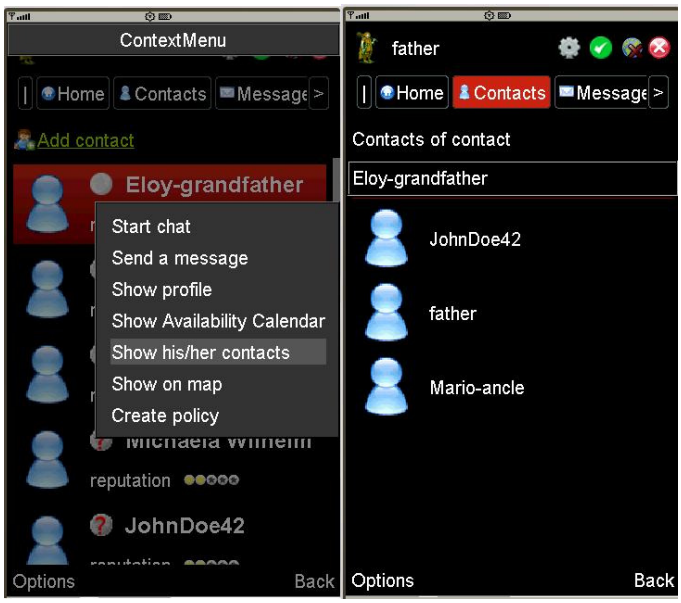


Figure 7: Share contact list screen.

2.2.4 Policy Manager

The privacy manager component was improved in the Gamers prototype specifically because the first trial community (angler) reported severe usability problems to handle this feature properly. In order to help the end-user to handle the privacy rules creation through subsequent steps. The new Privacy Manager component now is based on five consecutive screens (in contrast to one screen in the angler application):

- Screen 1: this is the Privacy Manager main screen, where the end-user selects the rule type: presence, location, profile or contacts.
- Screen 2: this screen will display the user's identities (PIDs) to select those to attach the rule to.
- Screen 3: in this screen the end-user selects the target users from the contact list affected by this rule: everyone, sub-community or a single contact.
- Screen 4: in the following screen the user defines the rule effect: "allow", "disallow", "ask once".
- Screen 5: the final screen shows a short summary of the created rule.

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

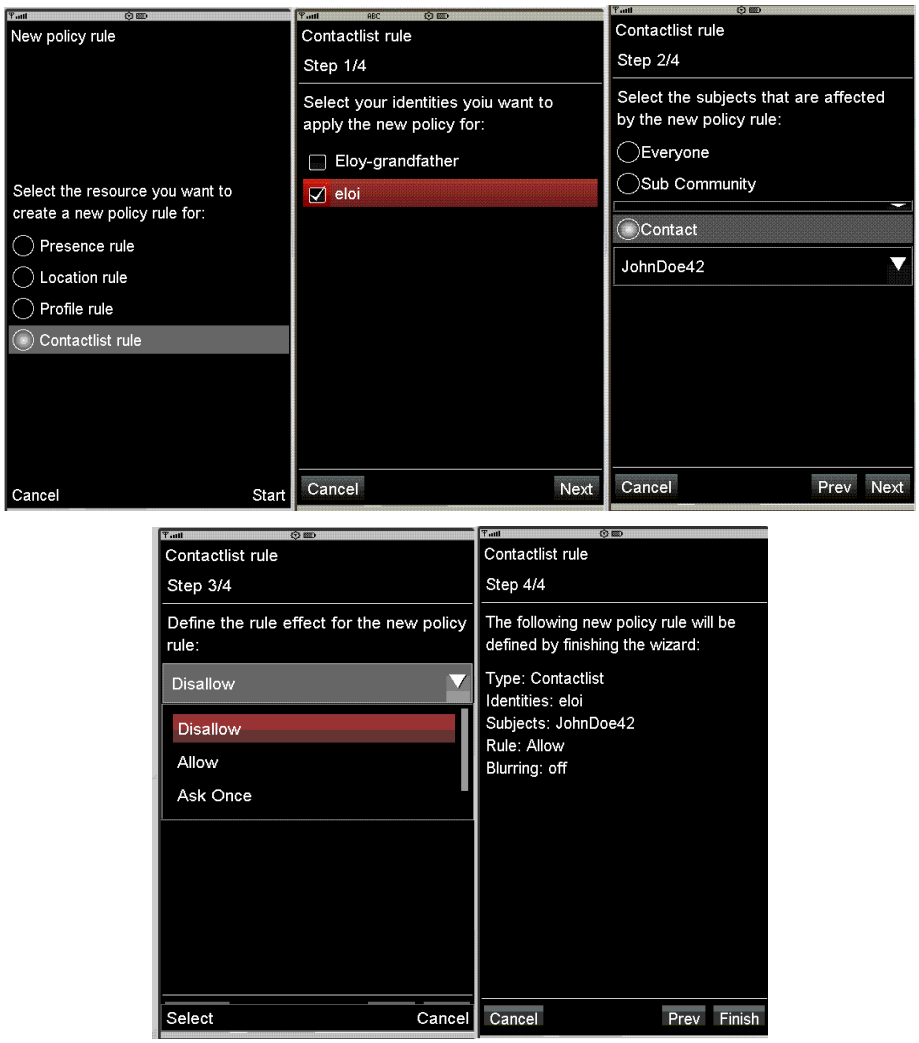


Figure 8: Privacy Manager screens

2.2.5 Privacy Advisor

Due to the increase of attributes in the profile, the user's privacy can be more easily put at risk. Therefore the scanning function of the Privacy Advisor (PA) has been extended.

For Gamers prototype the PA scans the information appearing in the following profile attributes "family name", "locality"," street name"," phone number", "zip code"," email", "Skype-ID", and "Facebook". These attributes are checked inside the following content types:

- Category content (in the file description, and inside a plain text file)
- Forum-thread contributions, and inside a plain text file attached to the post.
- Chat messages
- Asynchronous messages

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

2.2.6 Public Community

2.2.6.1 Public Community main screen

The public community is accessible from the tab "Community" in the navigation bar.

The main public community screen is shown below. The main difference with regard to anglers application is that all links related to anglers functionality (watercourses, species summary) were removed. Links to the public forum and the public repository remained.

Further there are new links related to public points of interests (see Section 2.2.10.2) and nearby users (see Section 2.2.10.3).

Figure 9: Public Community main screen

2.2.6.2 Access restriction to published content in the Public Community

In former versions of the application, all members of the community had access to the content in the public community. In gamers prototype an additional level of control by the user on his own content threads within fora) is introduced, according to the requirements collected by the gamers. The user is allowed to restrict when and who has access to the shared information by defining a rule at the creation point. This mechanism also implies more flexibility in the content sharing as it is not necessarily tied to publication within a specific sub-community.

The following pictures show the screens for the new rule creation.

The user will specify a subject for a thread, those contacts from his contact list that will be granted access ("Restrict Access To") and a period of time ("From" and "To" dates) by means of calendars windows for content availability.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

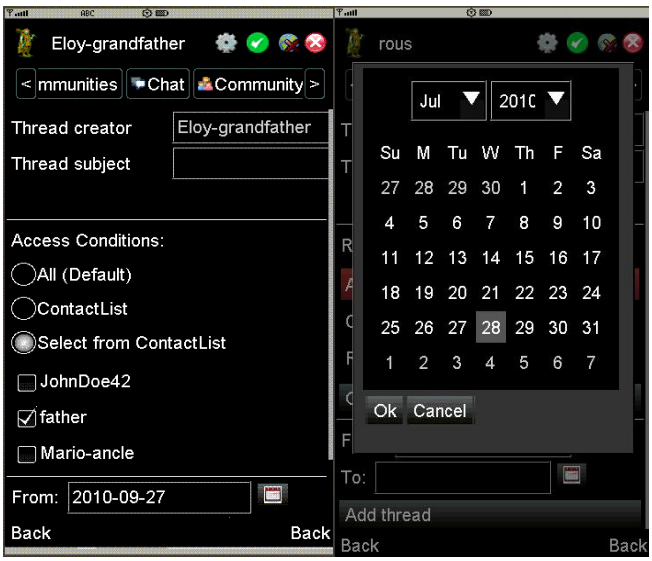


Figure 10: Rule creation attached to content in the public community screen.

The user might want to modify or remove any rule attached to a specific thread. With this purpose, the new application provides the option "Edit thread rule" that includes two buttons into the rule details screen: "Change Rule", "Delete Rule".

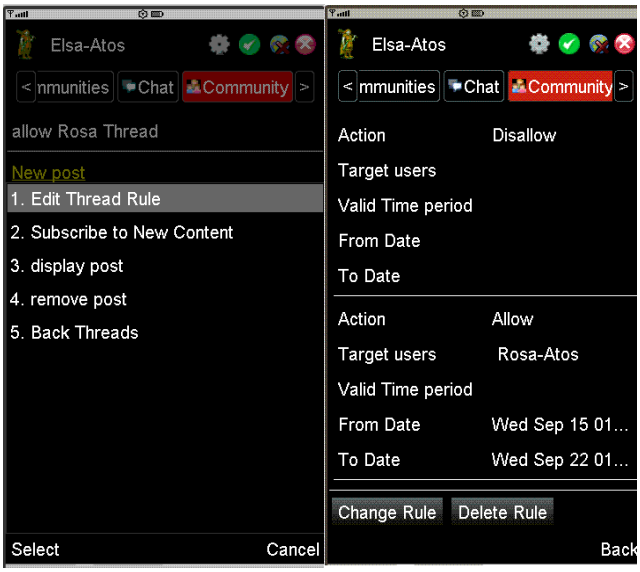


Figure 11: Edition of content rules in the public community.

2.2.6.3 Content access history

Each content contribution within the public community (a thread within a forum) stores an access record with partial IDs and access date of those members who accessed that content. This new feature allows the user to visualize the mentioned list.

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

The following image shows the option "Thread Access History" associated to a thread within the "Travian Forum" resulting in a screen with a counter of visitors and their partial IDs.

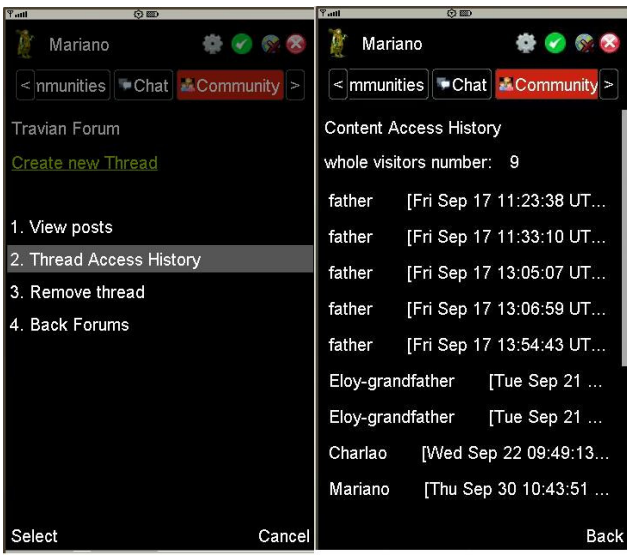


Figure 12: Access history in public community.

2.2.6.4 Notification for new available content

This feature allows end-user to subscribe to one thread in the fora of the public community. The user will receive a notification from the WP5 platform when a new post is available in the mentioned thread. The right screenshot of Figure 11 shows the notification when a thread ("Cousins" in the example) has received a new contribution. The screenshot on the left shows the options "Subscribe to new content" and "Unsubscribe to new content" that allows the user to manage the reception of such notifications.

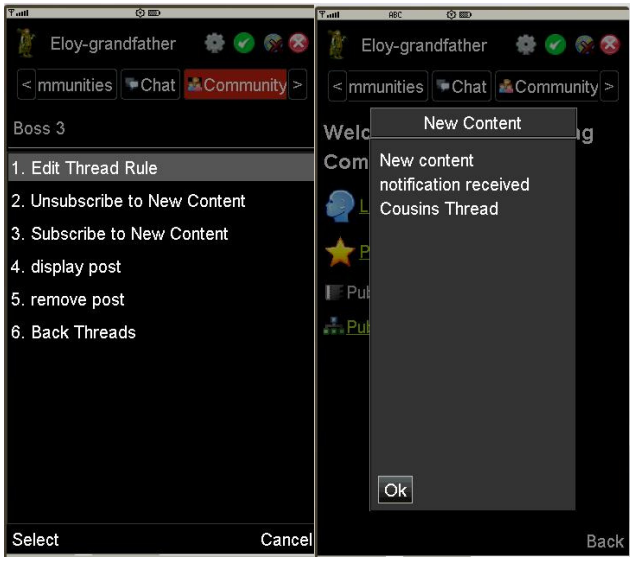


Figure 13: Notification for new available content in the public community.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

2.2.7 Public Repository

2.2.7.1 Access restriction to published content in the Public Repository

PICOS Public Repository is split in several folders or categories, and each of these categories contains some content, such as files or pictures. Similarly to section 2.2.6.2, this new feature allows the user to attach privacy rules to each category or file during its creation process. This offers the user an additional control level on the access to his published content in the repository by defining a rule that allows a person or set of contacts to get access to it during a specified period of time.

The following figure shows the screen for content creation. Along with a title, description and keywords for the content, the user will include a set of permitted contacts ("Restrict Access To") and a valid period of time ("From" and "To" dates) by means of calendars windows.

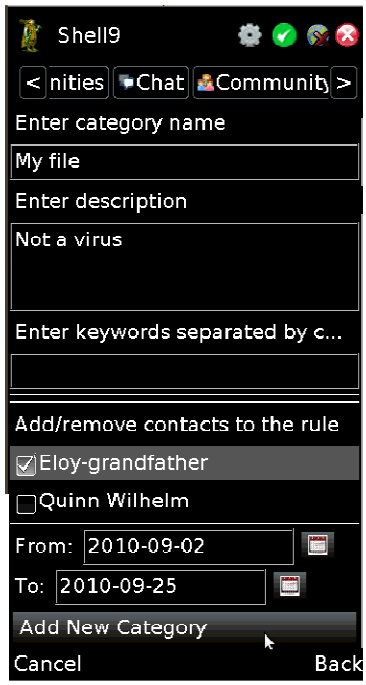


Figure 14: Rule creation attached to content in the public repository

The owner of a category can visualize the attached rules and modify them too. The following image shows the existence of a button "Show/Hide rules" to expand and display all the rules associated to a selected category.

In a further step the user has the possibility to modify or delete each rule by means of buttons "Edit" or "Remove". In the example there are two rules associated to the same category. Both allow the user "Carlos-atos" to browse the category, but at different periods of time.

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

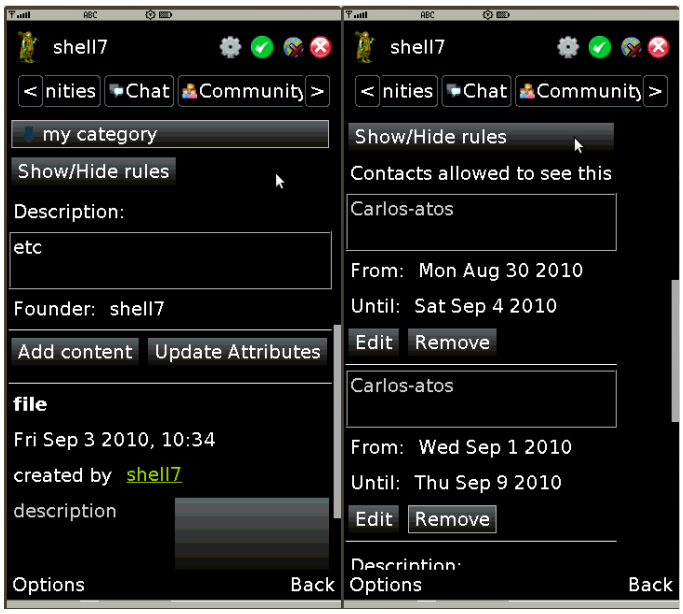


Figure 15: Display of rules attached to a category in the public repository.

The following image shows the removal of the second rule after clicking on the "Remove" button. The application asks the user for confirmation and indicates the successful operation.

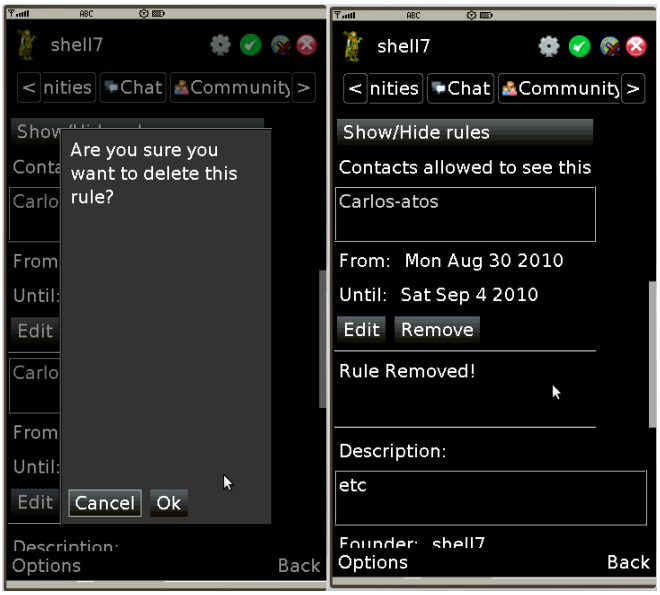


Figure 16: Deletion of a rule attached to a category in the public repository.

The user can modify the remaining rule by clicking on "Edit" button in order to either add/remove contacts allowed to see the category, or changing the limiting dates, and then clicking on "Save". The result is a modified category that applies to two contacts ("Shell8" and "Carlos-atos") for a different period of time.

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

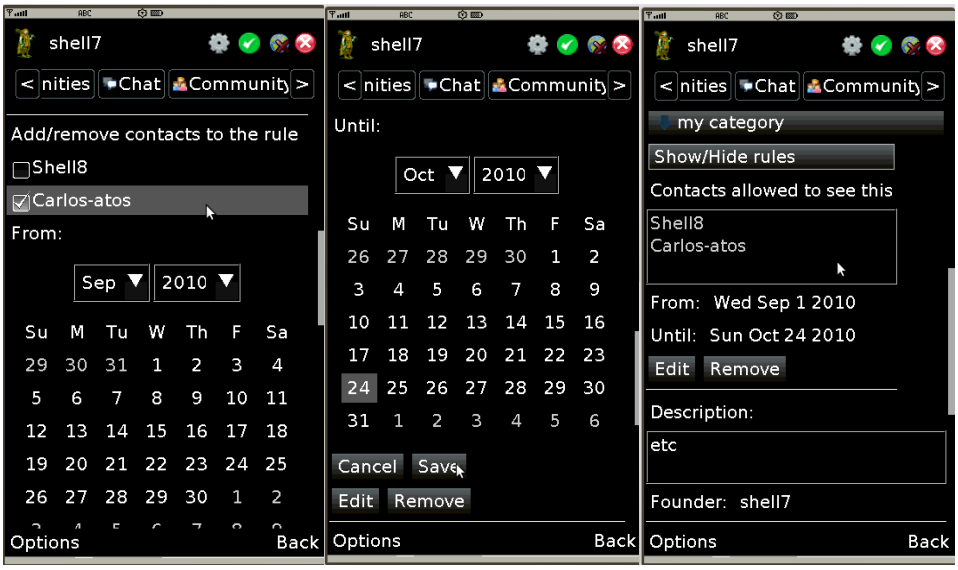


Figure 17: Modification of a rule attached to a category of the public repository.

2.2.7.2 Content access history

Similar to the access history in the public community forums, the content in the repository has the option to show the access history as well.

The figure below shows a button "Access history" to retrieve the access log associated to the content. This log indicates the number of visitors and provides a list of PIDs and the date of access in little endian format (starting with the day, hour in Coordinated Universal Time –UTC)

The button "Rate&Comment" on the right allows a member of the community to mark the content and add a comment, as long as he is not the creator of such content. Otherwise a button "Delete" is displayed.

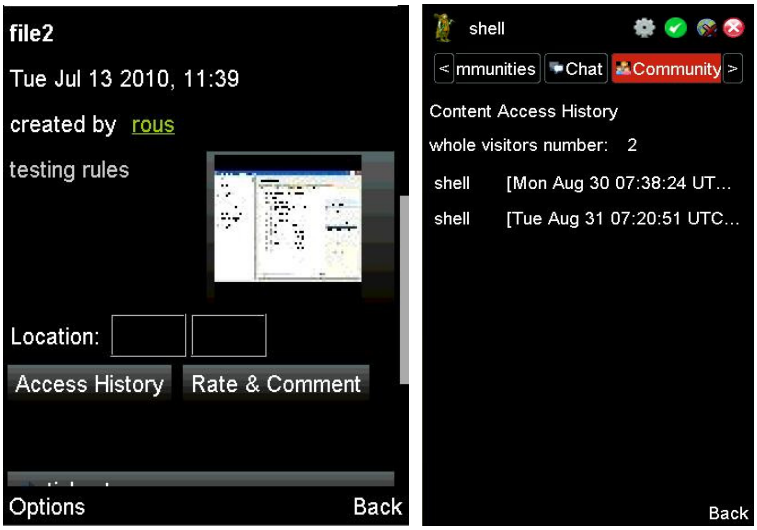


Figure 18: Access history in public repository

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

2.2.7.3 Notification for new available content

Similarly to the functionality implemented in public community, the user can subscribe/unsubscribe to notification mechanism from categories. This means, a subscribed user will receive a notification from the WP5 platform when a new file is uploaded to a specific category (text files and images).

The figure below shows the category "New" screen after the user's subscription to the notification mechanism.

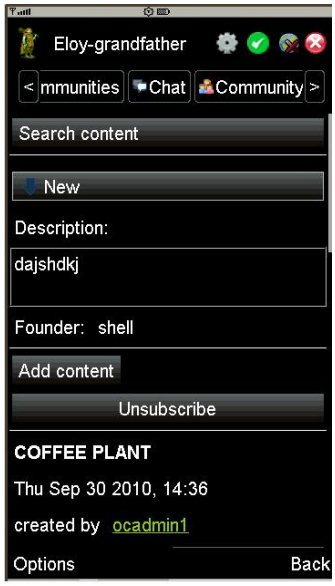


Figure 19: Subscribe to notification for new available content in the public repository.

2.2.8 My Files

The Private Room is accessible from the tab "My Files" in the navigation bar.

The main difference with the anglers application is that all functionality associated to anglers is dropped (for example catch reports). In this second prototype the user is able to upload a new file, change the file details or delete it.

The following picture shows the process of creation of a new file ("dependencias.jpg") within "My Files".

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Figure 20: My Files screen and example of file creation.

An existing or new file can be later transferred to:

- Private Sub-Communities.
- Public Sub-Communities.
- Public Community.

In the example below an existing file is transferred to the repository of the public community. The user is able to specify the category to transfer the file to, as well as allowed contacts and a valid period of time in which the file will be accessible.

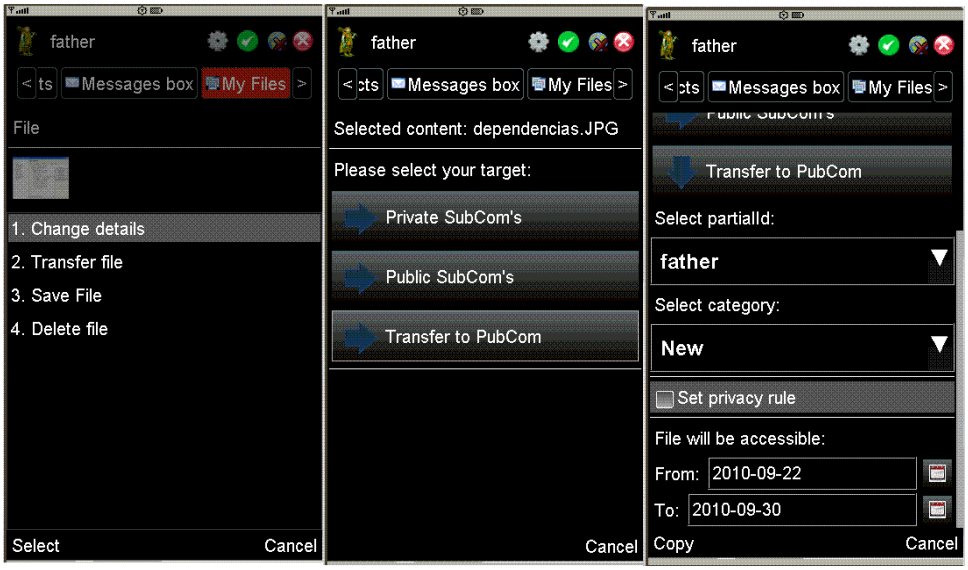


Figure 21: File transfer from My Files to Public Community

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

2.2.9 Sub-Communities

2.2.9.1 Access restriction to published content in Sub-Communities

This functionality is similar to 2.2.6.2 and implies a finer control of the user's own content within the sub-communicty. The user can manage his content published in the sub-communities (threads) defining a rule at the creation point that allows an individual or set of contacts to get access to it during a specified period of time.

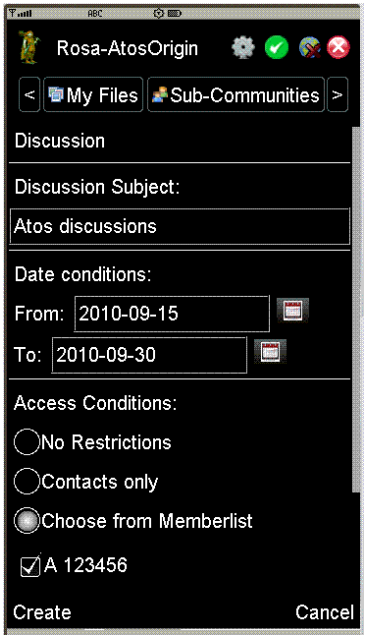


Figure 22: Rule creation attached to content in the sub-community screen

2.2.9.2 Content access history

This functionality is similar to the one described for the public community (see section 2.2.6.3). It only affects threads of forums of any sub-community. Nevertheless it differs in the fact that in sub-communities only the creator of the thread is allowed to see the list of partial IDs who had access to the specific thread and the time when they did it.

Figure 24 shows the sub-community thread "Atos thread" and how its owner retrieves such a list by using the "Options" command at the bottom of the terminal by clicking on "View Access History".

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

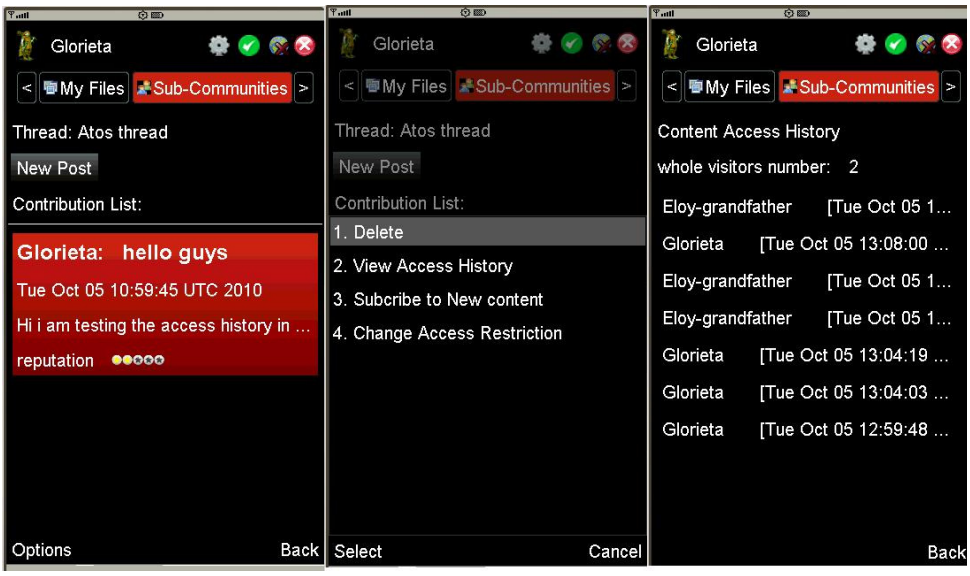


Figure 23: Content access history in Sub-communities.

2.2.9.3 Notification for new available content

This functionality is similar to the feature implemented for threads within fora of the Public Community. It allows the end-user to receive a notification from the WP5 platform informing him about the publication of new content in a thread of a sub-community. Figure 18 shows the option "Subscribe to new content" (screenshot on the left) for sub-communities (thread "Discussion Test" in the example). On the right, the screenshot illustrates the received new content notification when a new contribution is submitted.

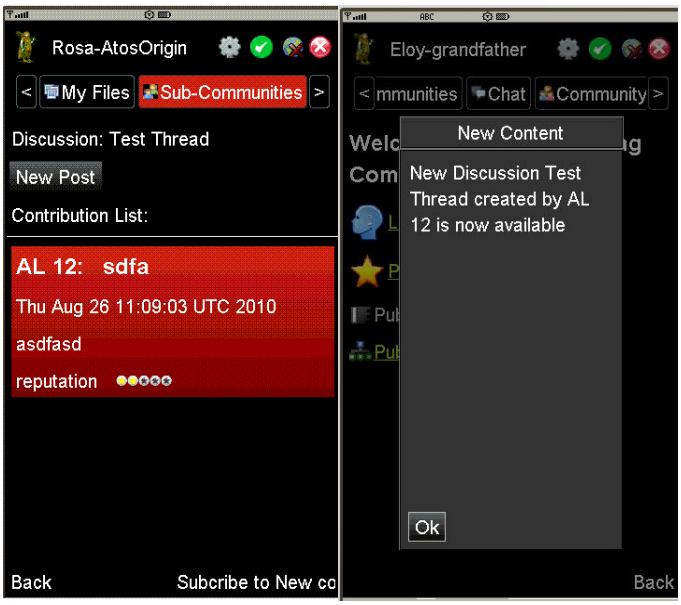


Figure 24: Access history in Sub-communities

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

2.2.10 Location Based services

Location Based Services (LBS) describe features that are related to the location of an entity (a user, a landmark or a point of interest). The location may be described as map coordinates (latitude, longitude) or as a street address (country, zip, town ...). These services are accessible from the tab "Map" in the navigation bar.

In anglers prototype LBS services consisted of features such as "My location", "Contacts location", "Watercourses" and "Fishing spots". "Watercourses" and "Fishing spots" disappear in Gamers prototype, since they are specific services for the angler community. However Gamers prototype counts on new location-based features described in the following sections.

2.2.10.1 Social Presence Awareness

The present feature automates the blurring settings around sensitive locations defined by the user, called "Private Sites". Examples of private sites can be home, work, etc. The user can define these private sites on a map with GPS coordinates and attach rules to these areas in terms of contacts requesting location information and range conditions. In case a contact wants to access the location information of the user, the server will check if the user is close to a private site, will evaluate the rule attached to the private site and apply the rule, as configured by the user (contact and blurring settings).

The following figure shows the creation of a new Private Site. First of all he should select "Add Private Site" from "Options" command and afterwards fill out the following fields: title, description, location coordinates (either by entering coordinates manually or by as drawing an area with the stylus on the Google map) and the private site size.

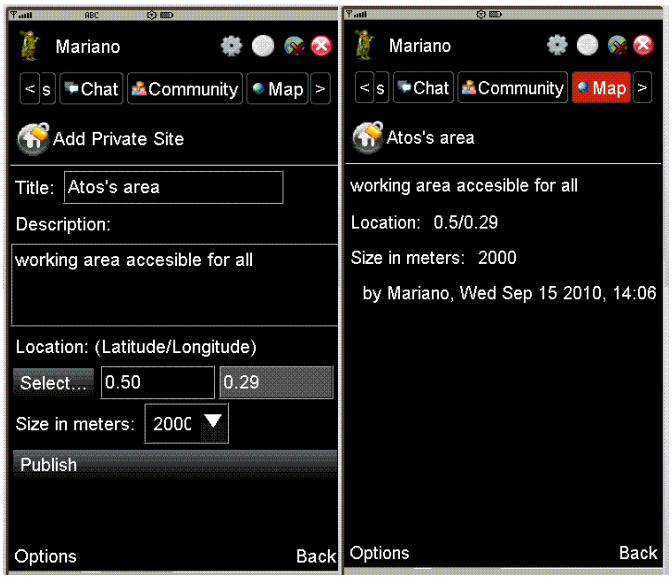


Figure 25: Creation of a Private Site.

A user will be able to visualize all his defined private sites by means of the "Map" option in the navigation bar (screenshot on the left) or by the "Options" command at the bottom of the terminal screen (screenshot in the middle).

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

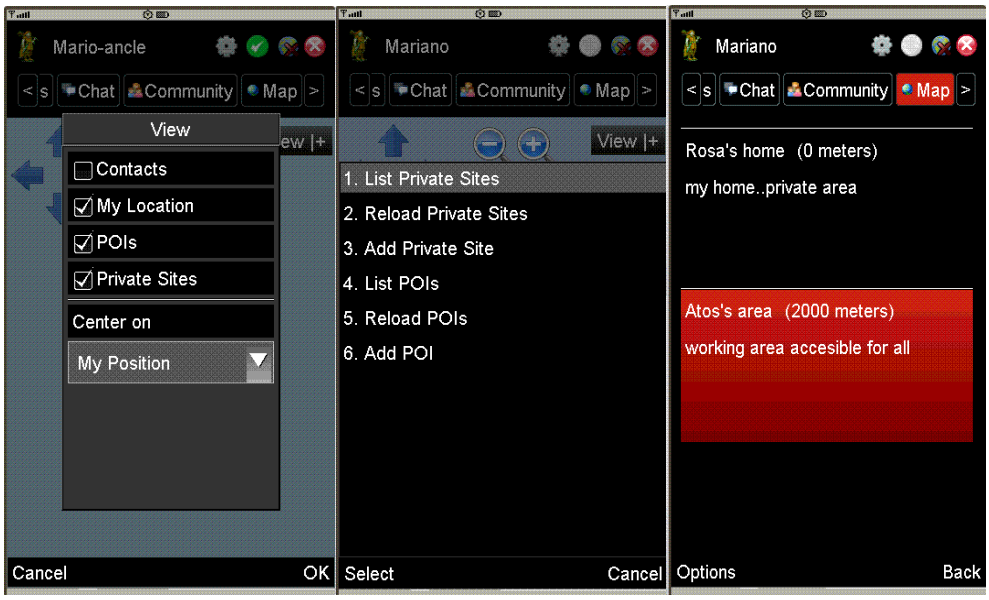
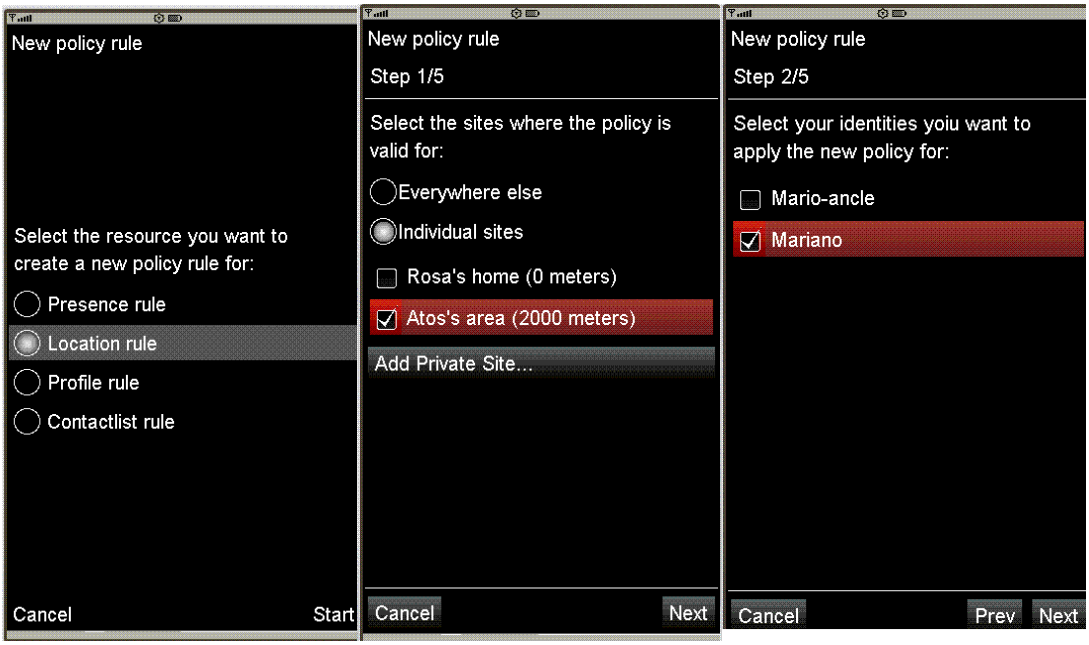


Figure 26: List Private Sites.

After creating a private site, the user can attach a policy rule to this item, in order to define which contacts will have permission to receive his location information when he is close to that position. This functionality is available in the new Privacy Manager version implemented in gamers prototype (See 2.2.4 for further information about the Privacy Manager).



Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

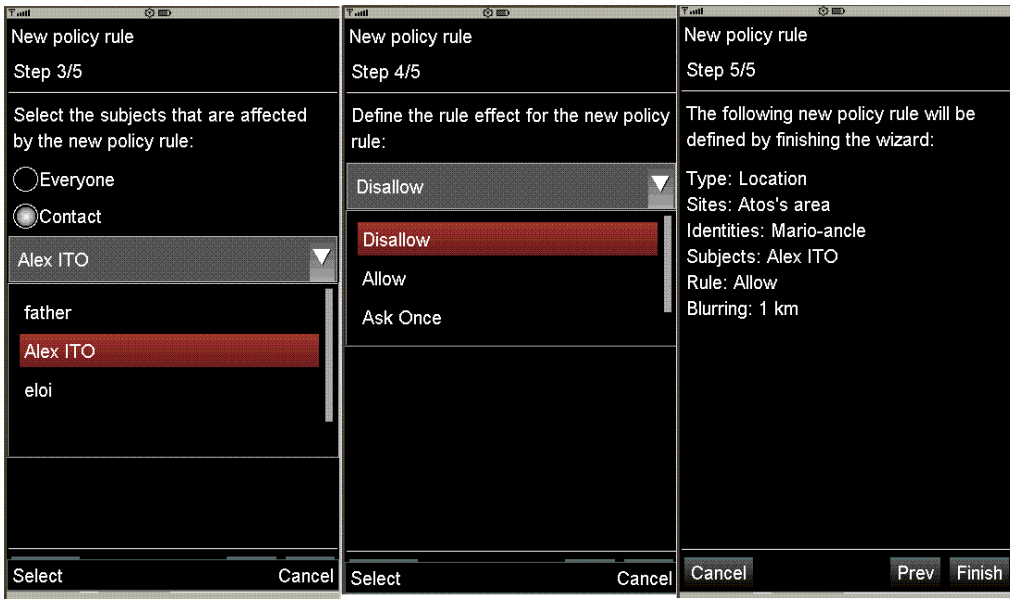
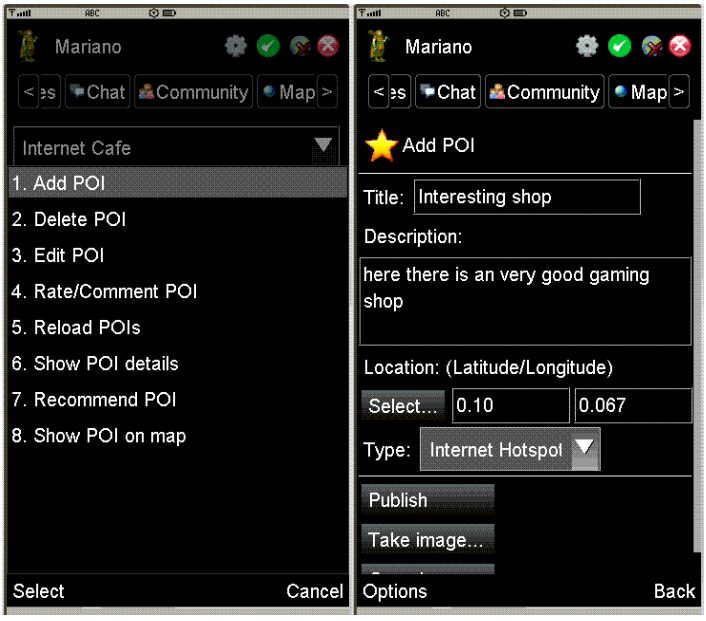


Figure 27: Definition of rules attached to Private Sites

2.2.10.2 Points of Interest

Likewise to private sites, the user is allowed to define, manage, display and create rules for interesting locations for a gamer (e.g. WiFi hotspot, Internet Café). These sites are called "Points of Interest" (POIs). The user can afterwards share or recommend these locations to other contacts. Rating of public POIs is also possible.

Figure 28 shows the creation process of a new POI from the "Options" command at the bottom of the terminal. The POI is described by a title, description, image, coordinates and visibility (public/private).



Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

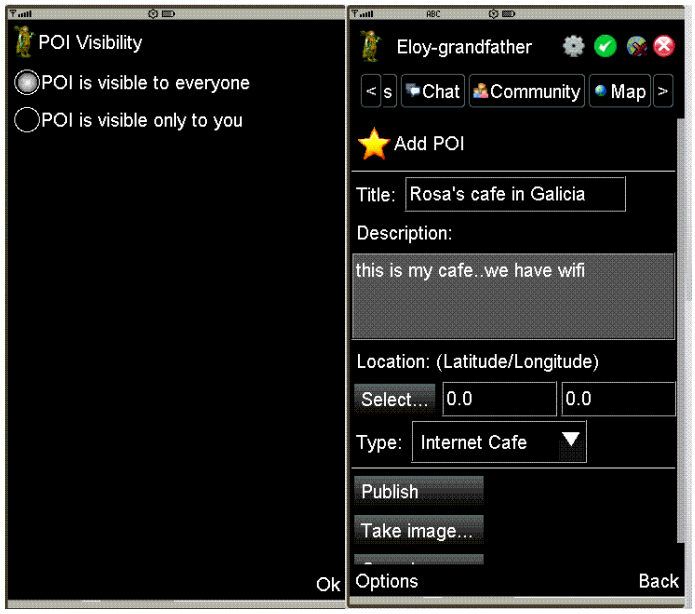


Figure 28: Creation of a new Point of Interest.

2.2.10.3 Meeting with nearby players

Starting from the "Community" or "Map" tab in the navigation bar the user can display other gamers who are close to his position. These gamers are not necessarily included in the contact list.

Figure 29: "List Nearby Users" link in "Community" tab.

The user may select the size of the area (in meters) around his own location in which the system searches for nearby users and gets a list of players that have activated their location sensor and allow their location to be retrieved by the current user.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Figure 30: "Nearby Users" screen.

The user may select multiple items from the list of nearby users and execute one of the available options: “Send message”, “Create contact”, “Start chat”, “Show on map”.

Figure 31: "Nearby Users"options.

- The "Send message..." option is only available when exactly one nearby user is selected. When clicked the user may enter a subject and a text and send a message to the selected nearby user.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community’s Seventh Framework Programme.

Figure 32: Send message to nearby user.

- The "Create contact...." option is only available when one or more nearby users are selected. When clicked new contacts are created for all of the selected nearby users.

Figure 33: Create contacts to nearby users.

- The "Start chat..." option is only available when at least one nearby user is selected. When clicked a new chat is started with all selected nearby users as participants.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Figure 34: Start chat with nearby users.

- The "Show on map..." option is only visible when the list contains at least one nearby user. The map is centred on the current user's location and contains icons for all nearby users that fit on the map (depending on the zoom level). By clicking on a nearby user the current user returns to the list of nearby users and the clicked nearby user is selected.

Figure 35: Show nearby users on map.

2.2.11 Real-time Content Sharing: Communication and Notification Means

2.2.11.1 File sharing in chats.

The chat is accessible from the tab “Chat” in the navigation bar. In gamers prototype a user can open a chat and share not only text messages but also files (i.e. images), as shown in the second message of Figure 36.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community’s Seventh Framework Programme.

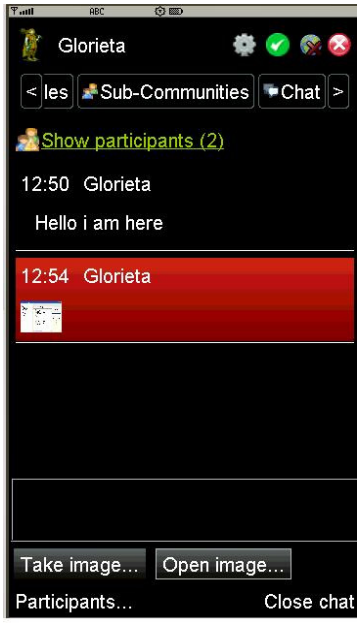


Figure 36: Real-time Content Sharing: file in chats.

2.2.11.2 Archive chat

Another extension of the chat functionality is the possibility to store the chats, so that the user is able to retrieve a list of the chats he has participated in.

Figure 37 shows the option "History" at the bottom of the Chat screen. When clicking on it the user is provided with a summary of the chats, including the date and participants.

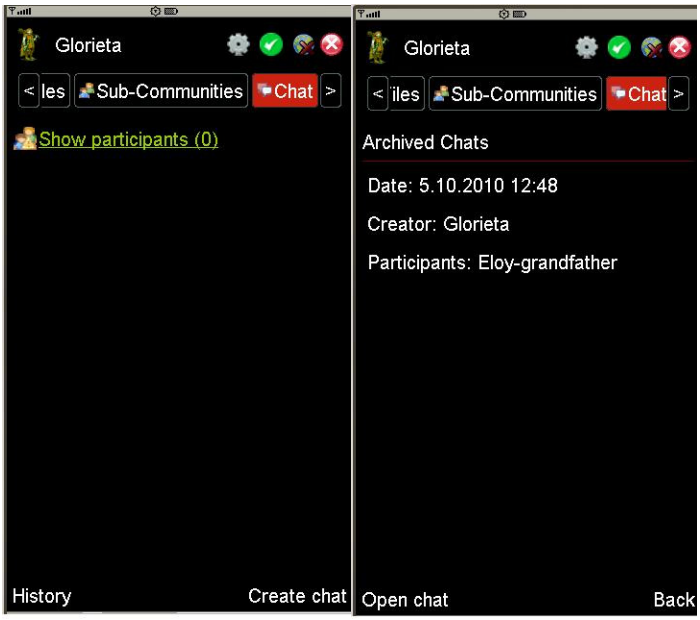


Figure 37: Real-time Content Sharing: archive chat

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

2.2.12 Notifications

The notification model implemented in the first prototype (Anglers Application) was not easy to handle, since it was required the end-user to be logged into the PICOS community to receive notifications from the server side. Notifications were not stored in the server but an asynchronous message was sent to the user with instructions to follow. This implied an inbox overloaded with messages. This was especially problematic for those notifications requesting authorization for displaying presence information.

This model changes in gamers prototype. The notifications are stored on the server while the user is offline. In the next login the client retrieves notifications from the server and displays them to the user. The set of notifications that this server mechanism handles comprise of:

- Authorization request for presence, location, contact list
- New content availability
- Sub-community invitation
- Sub-community change admin

2.2.13 Advertising services

This is a new feature allowing the creation of commercial advertisements and their reception. The advertising services can be described from two different perspectives:

- **Advertiser (commercial user).** Through a commercial account in the community, an advertiser can create a commercial advertisement. This advertisement is based on POIs, here called "commercial POIs" (cPOIs). cPOIs are a subset of POIs within the PICOS application. The cPOI is a POI that contains a target user profile. By matching the profile of users against the target user profile the WP5 server platform decides whether the advertising POI is suitable for a user or not.

  At the time of creation, the cPOIs are defined with a profile similar to an ordinary POI (see Figure 28), including:

  - Title
  - Description: commercial text.
  - Type of POI: Internet Hotspot or Internet Café.
  - Keywords
  - Location details
  - Image

  The second part of the cPOI screen permits the advertiser to set the target user of the advertisement, segmenting PICOS community by profile attributes:

  - Age, defined by a range between "ageMax" and "ageMin" (e.g. "from 20 to 40").

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

- Gender: a drop-Down-List with predefined values: unknown, male only, female only, male and female.
- Hobbies.

Figure 38. shows the mentioned target user screen (screenshot on the left) and the resulting cPOI (screenshot on the right).

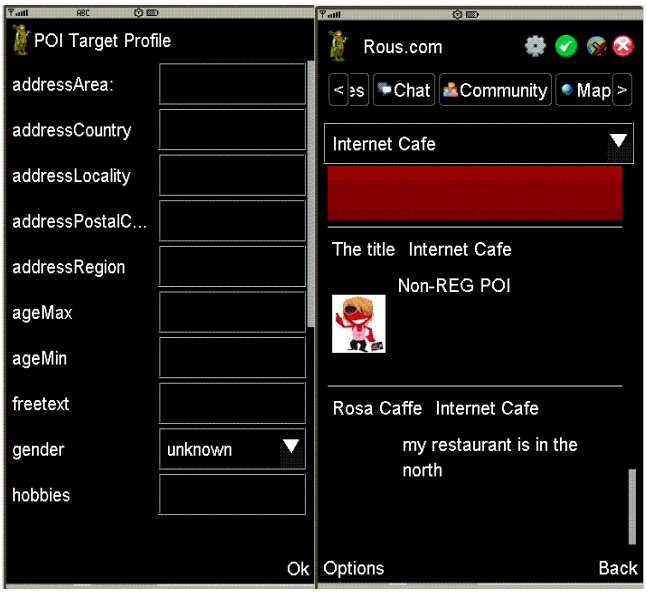


Figure 38. Commercial Point of Interest screen

- **PICOS prototype user.** A member of PICOS community will receive advertisements if he is subscribed to this service. The user will receive an advertisement if he approaches the location of the cPOI and his profile attributes matches those of the target user.

  Then the user can locate the cPOI on a map (equally to another POI). Further, the user can recommend the cPOI to his contacts (or a subset of contacts).

2.2.14 Revocation

Revocation occurs when a member leaves a community. When this happens PICOS platform retains the content/files/posts uploaded by that user in the community for other members to keep access to it, but it replaces the partial ID of the no longer existing owner by the label “anonymous”.

Figure 39 shows a link “Deactivate account” in the root profile of the user “Mariano” (screenshot on the left) and a confirmation message from the system (screenshot in the middle) before performing the account deletion. The screenshot on the right shows that another member of the community would see the mentioned label “anonymous” when accessing a file created by the revoked account.

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community’s Seventh Framework Programme.

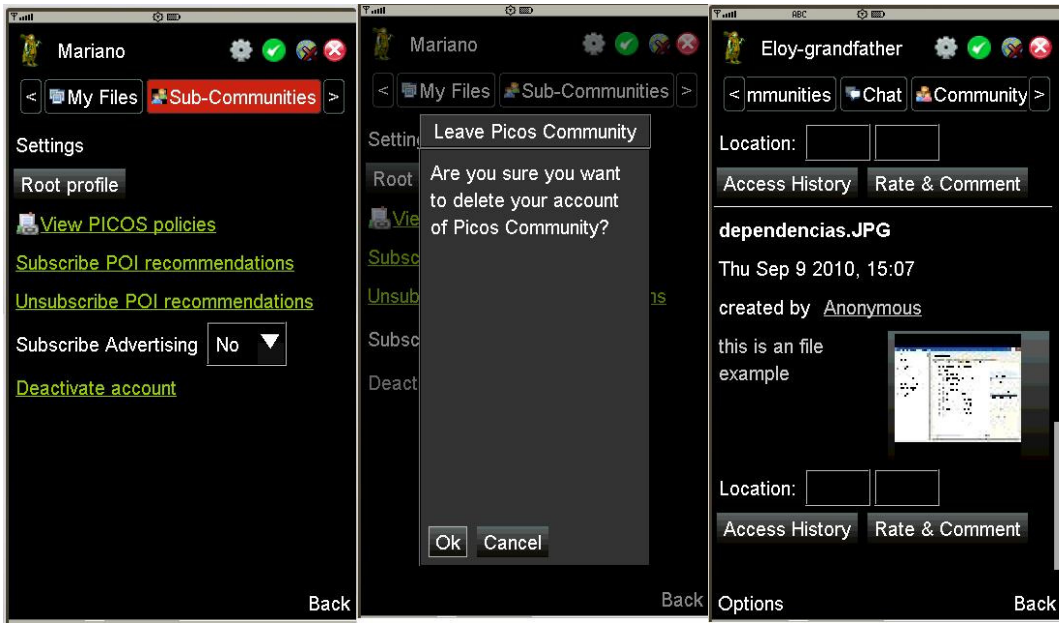


Figure 39. Revocation from PICOS community

2.3 *Usability improvements.*

Before the start of gamers trials (WP7), WP7 leader provided a usability report on the application screens to improve the usability and enrich the user experience for the prototype at the client-side.

The main changes applied were:

- Consistency of screens along the features.
- Layout changes to avoid overlooking of important functions (such as settings).
- New tabs in the navigation bar to ease access to features (e.g calendar).
- Message text and labels rewording to better indicate actions.
- Option commands reordering.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

3 High-Level Design of Gamers Client Architecture

3.1 Scope and purpose

This High Level Design (HLD) section describes the overall architecture of the Gamer Application from the WP6 perspective with its main components and their interfaces. Further, it describes how these components interact at a high level with each other.

3.2 Application Architecture Overview

The following figure shows the overall architecture of the Gamer Application:

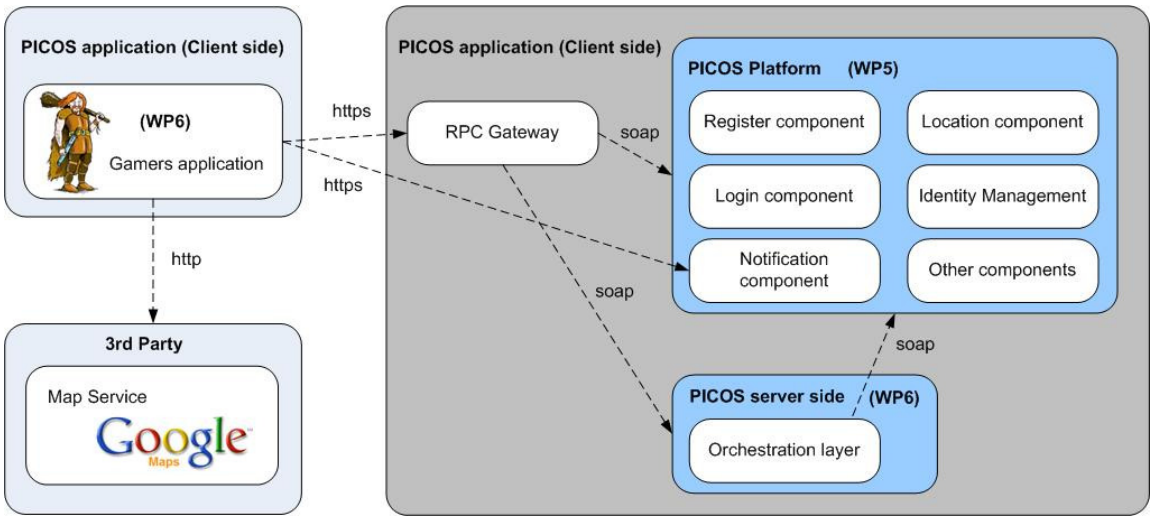


Figure 40: Overall Architectural Overview

As built on Anglers Application, the Gamer Application shows the same structure described in Section 2.1.2 of D6.1 [3], this is a client-server topology connected through a Remote Procedure Call (RPC) Gateway. The client side also communicates with a third party map service (Google Maps). The differences in the architecture between both prototypes lies in the disappearance of the Species Summary server from the WP6 server side and the Fishbase database from the third party block, which are related to the angler community but are not necessary for the gamers.

In the following, we describe shortly the mentioned components.

3.2.1.1 PICOS WP6 Client Side

Similarly to Anglers prototype, the PICOS WP6 Client side uses a Nokia 5800 as a hardware platform and the J2ME (Java 2 Mobile Edition) environment with the installed PICOS Gamer Application. This application communicates via https with the RPC Gateway and via http with a third party map service.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Two connections are used between the PICOS client side and the PICOS server side: One for client initiated events (e.g. login to the platform, retrieve contacts, send messages) and one for server side initiated events (e.g. notifications for new chat messages or advices from the Privacy Advisor).

The PICOS Angler Application is structured in different layers as shown in D6.1- Section 2.1.3.1 following a Model-View-Controller (MVC) paradigm/design pattern.

3.2.1.2 PICOS WP6 Server Side

The WP6 Server Side consists of the Orchestration Layer, whose objective is to combine function calls to the WP5 platform and to perform a pre-processing on the server side to prepare responses which are adjusted to specific WP6 uses cases.

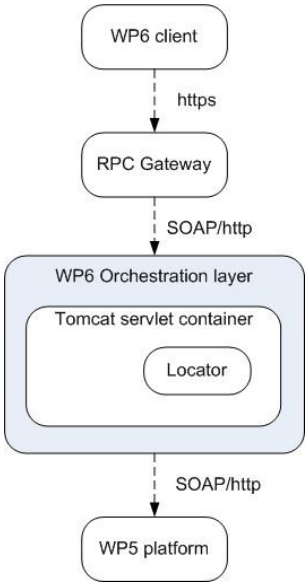


Figure 41: Orchestration layer in Gamers prototype

3.2.1.3 PICOS WP5 Platform

The WP5 Platform is responsible for all community agnostic services. The WP5 Platform consists of a variety of components, such as Location, Notification, Policy, Presence component, among others.

Beside the Notification Component, all components receive requests from the client side and response to the requests. In case of the Notification Component, the message flow is triggered by the component itself.

3.2.1.4 RPC Gateway

The RPC Gateway is part of the WP5 Platform deliverable and unifies the access from the client to the server. In combination with the RPC library on the client side, the messages are encoded/decoded and transferred.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

3.2.1.5 3rd Party Map Service

For the location-based services Google Maps is used as a map service to display maps with the locations of the user himself, his contacts, private sites, points of interests and gamers nearby.

3.2.2 General Functions

No changes with regard D6.1. [3]

3.2.3 Non-Functional Aspects

No changes with regard D6.1.

3.2.4 Development Tools and Libraries

No changes with regard D6.1.

3.2.5 Development Environment

No changes with regard D6.1

3.2.6 Deployment

No changes with regard D6.1

The latest version of the gamers application (Jad and Jar files) can be downloaded from the following Over the Air (OTA) server: https://193.254.175.105:7020/otas

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

4 Gamer application prototype Design

Complementing the description of the Gamer application from the user perspective, this section provides a wider and technical description of the referred features by means of class and sequence diagrams, following the MVC approach. Every feature is also associated to its correspondent requirement in the internal deliverable R2 [1].

4.1 Gamers profile manager

The changes in the user profile for this second prototype (Gamers application) refer basically to new added fields, and others removed, from those specific fields for anglers.

As mentioned before, the insertion of hobbies is different to the anglers prototype. The reason for this change is the advertising service feature that checks the attributes "hobbies" in the users' profile for matching. In this case the WP5 platform needs a separated list of objects instead of a comma-separated single field of hobbies. This change requires no additional classes, therefore the class diagram remains as displayed in D6.2a. The sequence diagram is also the same, with the slight difference that the profile class has a call to itself to add or remove fields to the screen on demand.

4.2 Presence and availability management

This functionality is related to requirement "REQ: Gamers R15: enriched status information". This new feature allows the user to set his availability status (or another status type, for instance: vacations, busy, lunch, working) and share this status with his contacts in order to arrange dates for playing or other appointments.

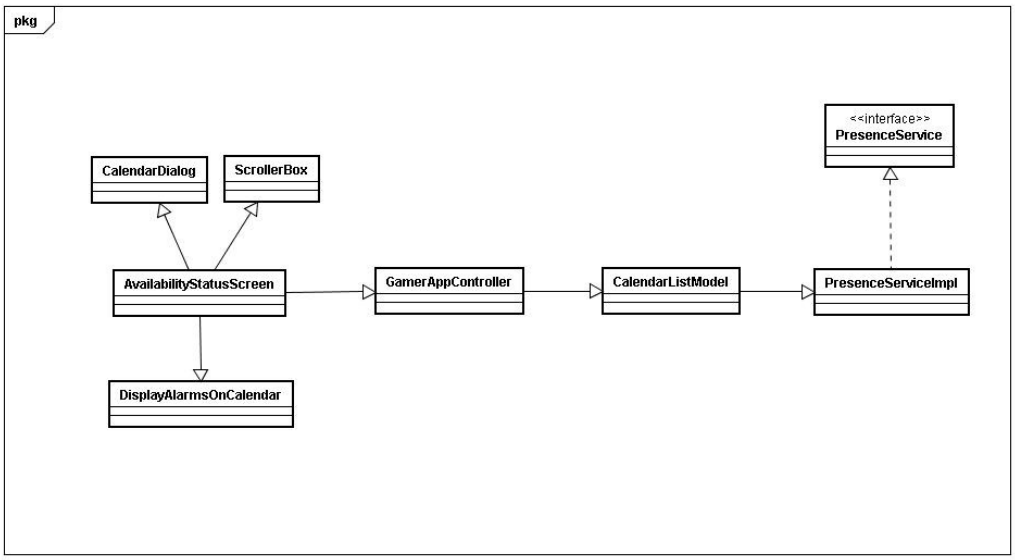


Figure 42: Calendar Class Diagram

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

The main involved classes in this feature are shown in the above picture:

- "`AvailabilityStatusScreen`" class: this is the main class which implements the option of displaying a calendar that shows the user's status. By invoking the class "`DisplayAlarmsOnCalendar`" the end-user is allowed to enter a new status invoking "`CalendarDialog`" for setting dates and invoking "`ScrollerBox` " for times.
- "`GamerAppController`" it is the controller class in charge of triggering the screens for the end-user. It also invokes the class "`CalendarListModel`", where the request/methods to the WP5 platform are implemented.
- "`CalendarListModel`" class implements the next methods:
  - "`fetchAvailabilityStatus`" for retrieving a selected contact's availability status from the platform.
  - "`fetchMyStatusses`" for retrieving the user's own status from the platform.
  - "`setAvailabilityStatus`" for setting a new status.
  - "`deleteAvailabiltiyStatus`" allows the user to delete a previous status.
- "`PresenceServiceImpl`" class implements the request/methods related to the calendar towards the platform.

4.3 Contacts management

This functionality is related to requirement "REQ: Gamers R1: Sharing Contact-List". As a new feature in the second prototype a user could inspect the address book of his contacts. This extends the contact list feature of first prototype. Figure 43 shows the possible use cases:

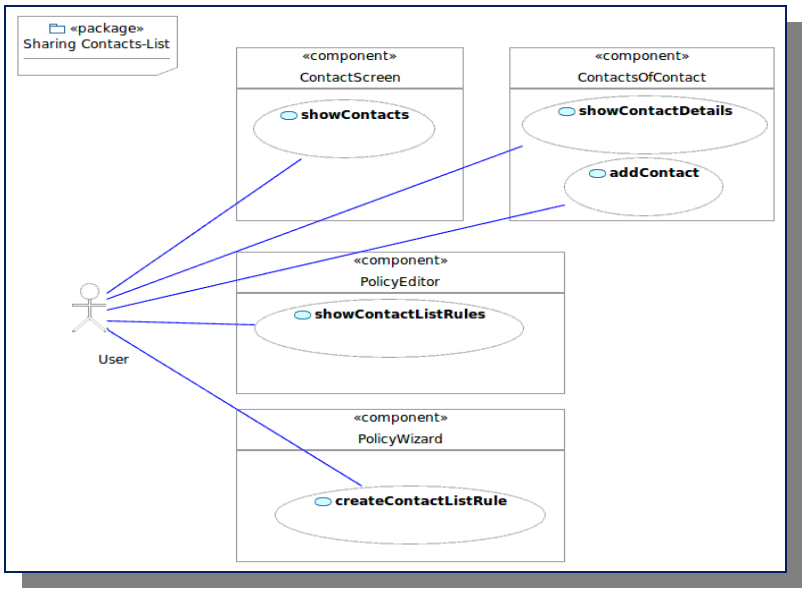


Figure 43: Use cases of sharing Contact-List

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

In detail the new contact management uses cases are:

- showContacts:
  The user could invoke a new command in the ContactsScreen to display the contact list of the current selected contact. When a user makes this request the first time the selected contact will be asked if he permits or denies this request. By default every user has a policy rule for the resource "contact-list" on "community-level" with the definition "ask once". Depending on the response of the requested user, the requesting user sees either a "access denied" message or a new screen (ContactsOfContact) containing the address book of the selected contact.
- showContactDetails:
  When inspecting the address book of a contact, the user could retrieve further details of a contact. The contact details page is the same like the detail page of the user's own address book.
- addContact:
  When inspecting the address book of a contact, the user could take a selected contact over to his own address book. If the given contact is already in his contact-list or if the user tries to add one of his own identities, he will get an accordant error message.
- createContactListRule:
  In the policy wizard (policy manager) the user could choose to create a manual policy rule in the contact lists of his identities. The rule could be defined for public-community, a specific sub-community or an individual contact. Further the policy could embrace multiple identities. As rule effect all standard values are available, like allow, deny, askOnce, askAlways.
- showContactListRules:
  The policy editor has a special tab page, where all created contact-list related policy rules are listed. Like any rule in the policy editor the user could edit or delete such a contact-list rule.

The use cases "showContactDetails" and "addContact" were already part of the angler application. The only difference is how the partial identity of the contact is retrieved. Therefore a deeper description with sequence and class diagrams are left out here.

The use case "createContactListRule" and "showContactListRules" are special cases of the new policy management implementation, which are described in detail in the following chapter.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

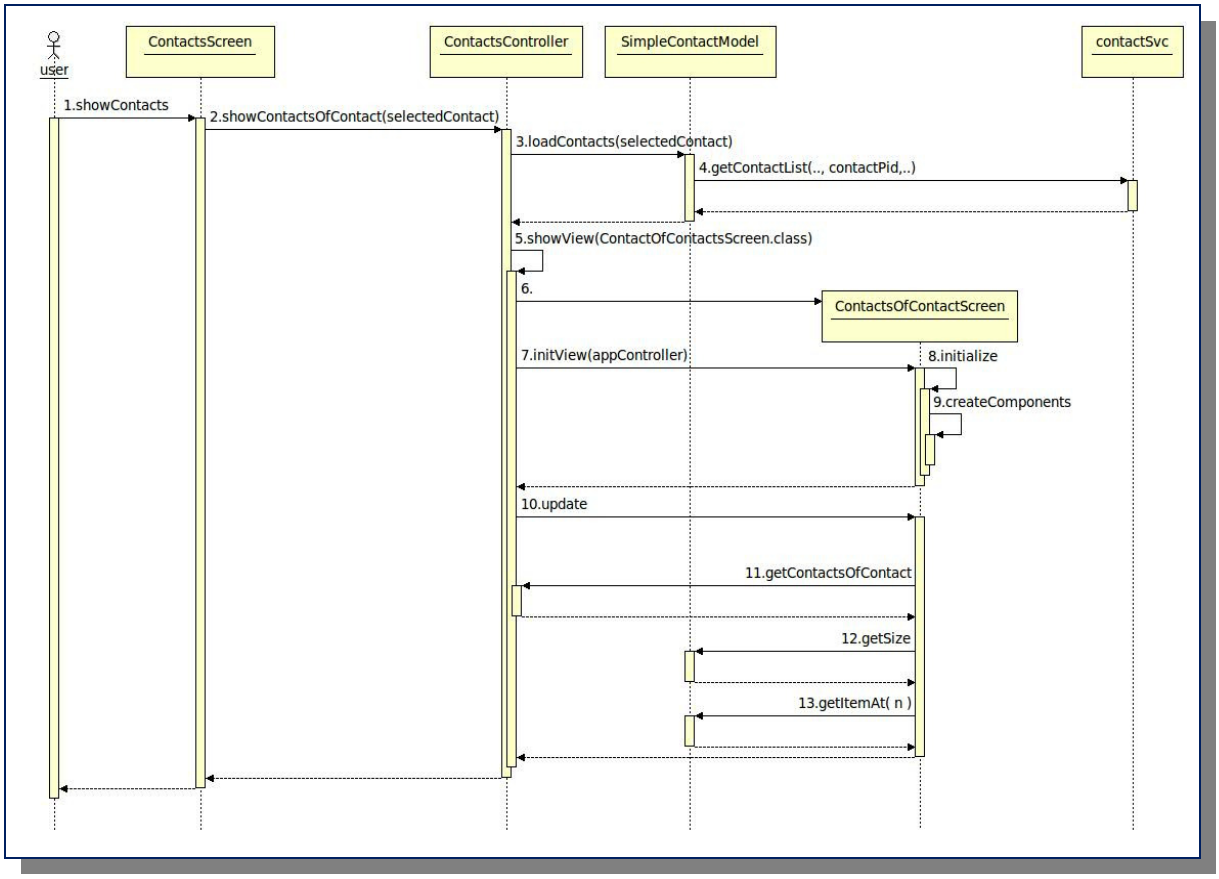


Figure 44: Sequence diagram "show contacts of contact"

A typically sequence for retrieving the address book of a contact is given in Figure 44. The diagram shows the method flow for a case where the contact has already given his consent to read his contact list.

1. The sequence starts when the user is in his contacts screen and selects the menu "show contacts" for a selected contact.
2. The request will be delegated from the `ContactScreen` to the method "`showContactsOfContact`" of the `ContactsController`.
3. The contacts controller has a new model, "`contactsOfContact`", of type "`SingleContactModel` ", which could hold a list of contacts separately from the "normal" address book. This model will be initialized through the method "`loadContacts`" and the partial identity of the contact as parameter.
4. The `SingleContactModel` fetches the contacts by invoking the WP5 method "`ContactService#getContactList`".

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

5. After initializing the "`contactsOfContact`" the ContactsController opens the view "`ContactsOfContact`", updates the data bindings and makes it the current visible view. This happens in the method "`ContactsController#showView`", which is described in the following:
6. First a new instance of the view class "`ContactsOfContactScreen`" is initialized
7. The reference to the application controller is passed through the method "`View#initView(Controller)`"
8. After storing the controller reference in the view object the abstract method "`View#initialize`" is called.
9. The method `ContactsOfContactScreen#initialize` initializes fields and creates the visible parts in "`createComponents`".
10. After creating and initializing the View object, the used model(s) is/are bound to the visible controls in "`View#update`".
11. The `ContactsOfContactScreen` retrieves the current contacts list from the `ContactsController` and passes this as data model to the contacts listbox.
12. The LWUIT list component calls `ListModel#getSize` and
13. `ListModel#getItemAt` to render the contacts on the current view.

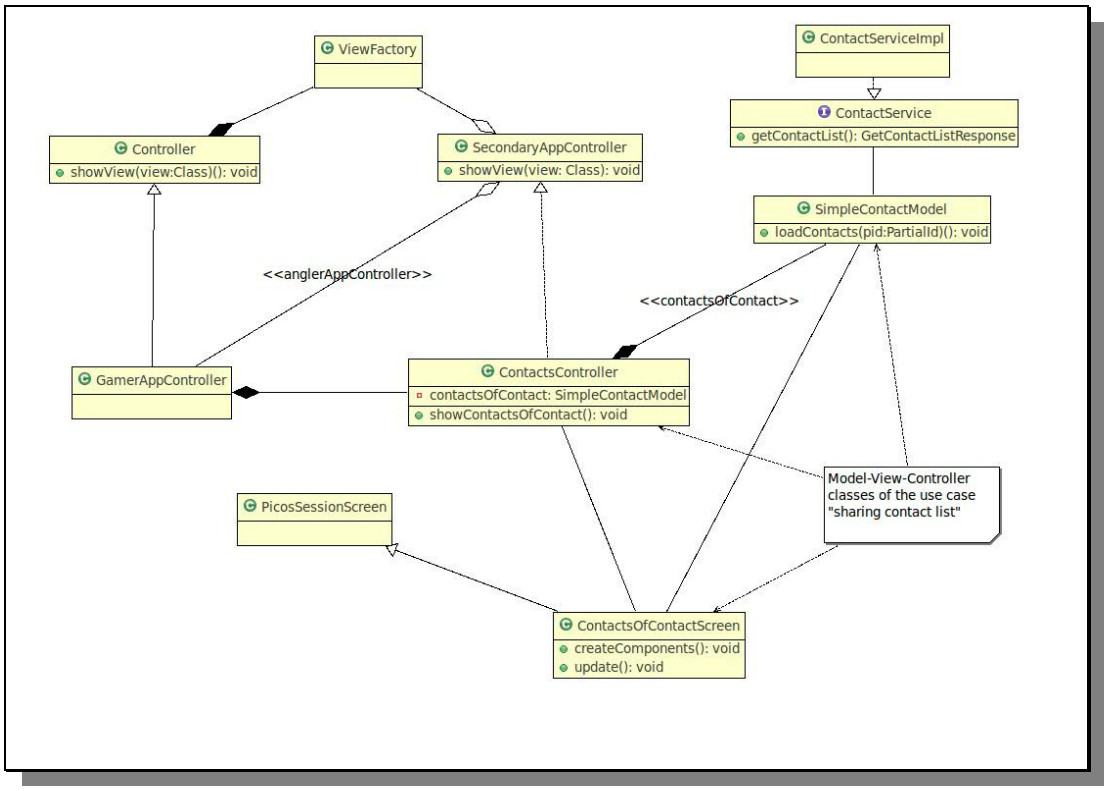


Figure 45: Class diagram "sharing contact list"

The used classes of the contact list management use cases are show in Figure 45. The main classes are `ContactsController`, `SimpleContactsModel` and `ContactsOfContactScreen`.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

4.4 Policy Manager

The policy management is completely rewritten in the gamer application. From birds eye view there are now two main components: A policy editor, which displays the user defined and non-system default policy rules, and a policy wizard, which guides a user through the generation of new policy rules.

While the policy editor doesn't embrace any real new concepts, the usage (and implementation) of multi step wizards is new to the gamer application.

The new policy wizard offers the user to create a new policy rule for one of the four resource types Location, Presence, Profile or Contact List.

Representative for the four possible wizard flows, the following describes the generation of a new policy rule for his contact lists.

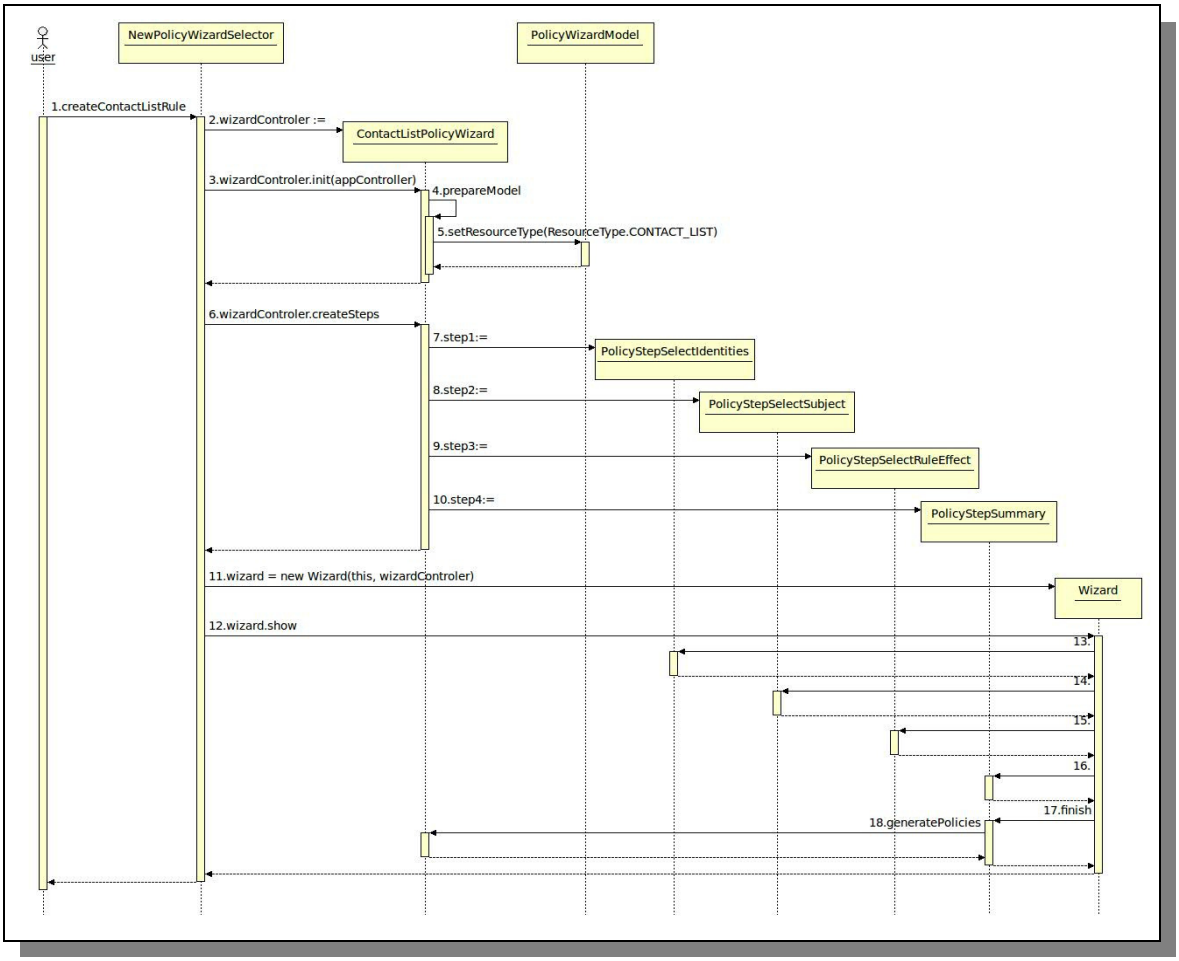


Figure 46: Sequence diagram "create contact list rule"

Figure 46 depicts a typical method flow of the new policy wizard.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

1. The starting point for a new policy rule is the `NewPolicyWizardSelector`, which offers the user one of four possible `PolicyWizards`. In the sample above the user has selected "new contact list rule" which triggers the method "`createContactListRule`".
2. The wizard selector creates a new instance of `ContactListPolicyWizard` as a special characteristic for a PolicyWizard.
3. Next the application controller reference will be passed to the wizard due invoking `WizardControler#init.`
4. The wizard contains - like all PolicyWizard descends - a reference to a policy model (from type `PolicyWizardModel`). As `ContactListPolicyWizard#init` is the first method and all necessary references are solved, the method `prepareModel` will be called here.
5. The only necessary initialization step is to specify the target resource type. All other attributes are kept or left untouched, to support operations like forward, backward, abort, resume without loosing all previous entered information.
6. Before presenting the wizard dialog to the user, all processing steps are calculated and initialized in "`WizardControler#createSteps`".
7. As first wizard step the user will be asked to specify at least one of his identities.
8. Now the user should select the policy subject: "public community", a specific "sub-community" or a specific contact.
9. In the third wizard step the user will be asked to specify the rule effect, like "permit", "deny", "askOnce", "askAlways".
10. The last wizard page is a summary page, where the user could double check the policy details before creating it.
11. This steps creates a new generic Wizard screen object as frame for the concrete wizard pages.
12. As next the wizard form will be presented to the user, where user could navigate through the single wizard steps.
13. Each wizard step creates the user components required for the current step and is able to manipulate the policy model.
14. The policy step "`PolicyStepSelectSubject`" takes use of the contacts- and subcommunity-model to avoid too many server calls.
15. The new policy rule will be prepared "in memory" without sending any (policy) request to the server.
16. Each policy step has a method to enable/disable the "finish button", as well as a corresponding method `PolicyStep#finish.`
17. In the current case only the `PolicyStepSummary` enables the finish button, which means that this step is also responsible to start the policy creation.
18. The summary step delegates the policy creation to the current `ContactListPolicyWizard`, to separate the user interface from the business logic. The `ContactListPolicyWizard` takes all

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

entered information out of the `PolicyWizardModel` and converts this into WP5 policies and policy rules.

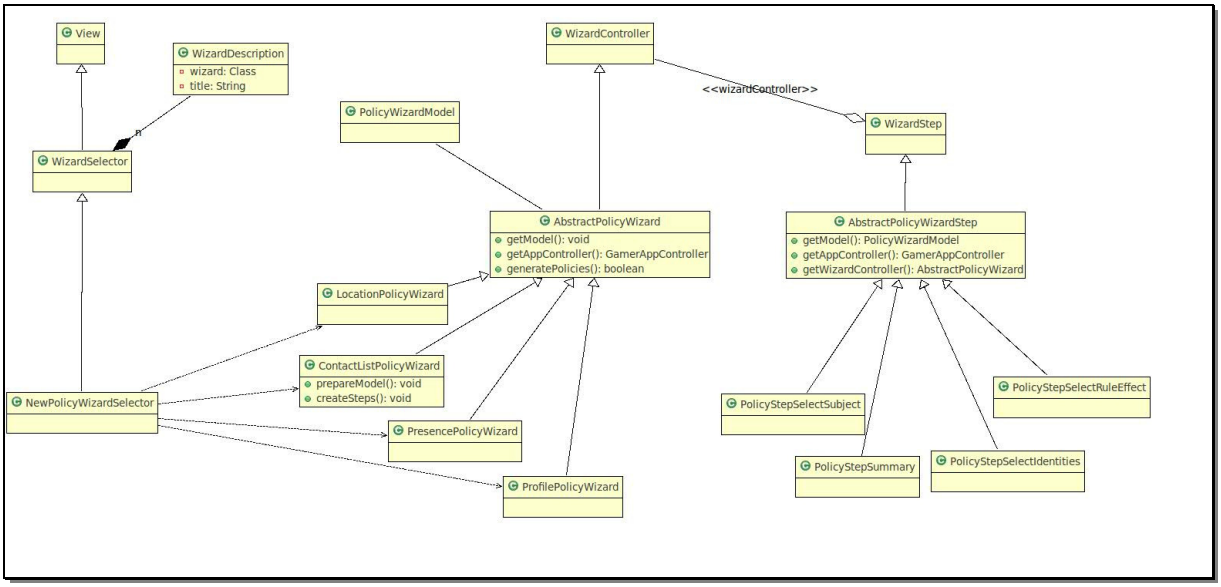


Figure 47: Class diagram "Privacy Manager"

Figure 47shows the classes, which are used by the new introduced policy wizard.

4.5 Privacy Advisor

This functionality is related to requirement "REQ: Anglers R34: Enhance Content Awareness".

The improvement related to this component only applies to the platform, so there is no impact at client level. The internal workflow of actions is not depicted as this belongs to the PICOS Platform side.

4.6 Public Community

4.6.1 Access restriction to published content in the Public Community

This functionality is related to requirements "REQ: Gamers R3: restrict access to published content" and "REQ: Gamers R4: restrict access based on date condition" applied to content (thread, posts) in the public community.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

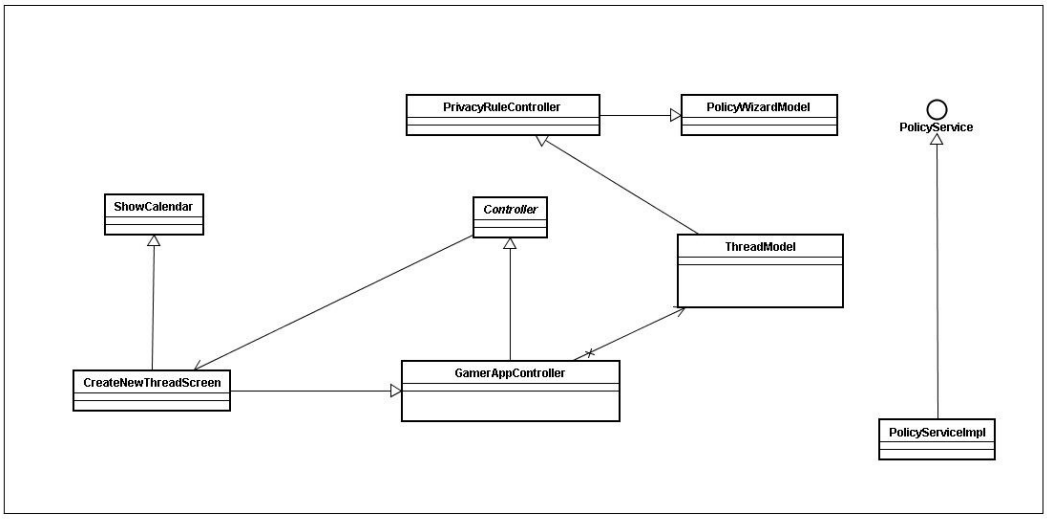


Figure 48: Access restriction to thread content (Posts) class diagram

As shown in Figure 48, the main involved classes in this scenario are the following:

- `"CreateNewThreadScreen":` this class implements the screen to add a new thread in the active forum. The end-user is able to enter a subject and a description for such thread, as well as to attach a privacy rule by defining those contacts from his contact list to which it applies.
- Moreover this class invokes the class `"ShowCalendar"` in order to display a calendar, allowing the user to specify a valid period of time (with start and finish date) for the privacy rule.
- `"GamerAppController":` it is the controller class in charge of triggering the screens (displaying the screens to the end-user). It also invokes the class `"model"`, in which the request/methods to the platform are implemented.
- `"PrivacyRuleController":` it is the controller class in charge of invoking the `"PolicyWizardModel"`, which contains the methods for adding and removing privacy rules.
- `"ThreadsModel":` this is the model in charge of managing the threads within a specific selected forum. It retrieves the thread list from the server side and contains the methods for adding/removing threads.
- `"PubliccommunityServiceImpl":` this is the class where the requests to the platform for managing the forum threads are implemented, such as: "retrieve the threads list", "create/remove/update thread", "retrieve the thread posts list", "upload/remove/change posts into the thread".
- `"PolicyServiceImpl":` this class implements the following requests to the platform side with regard to policies: "add", "remove", "modify" and "evaluate".

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

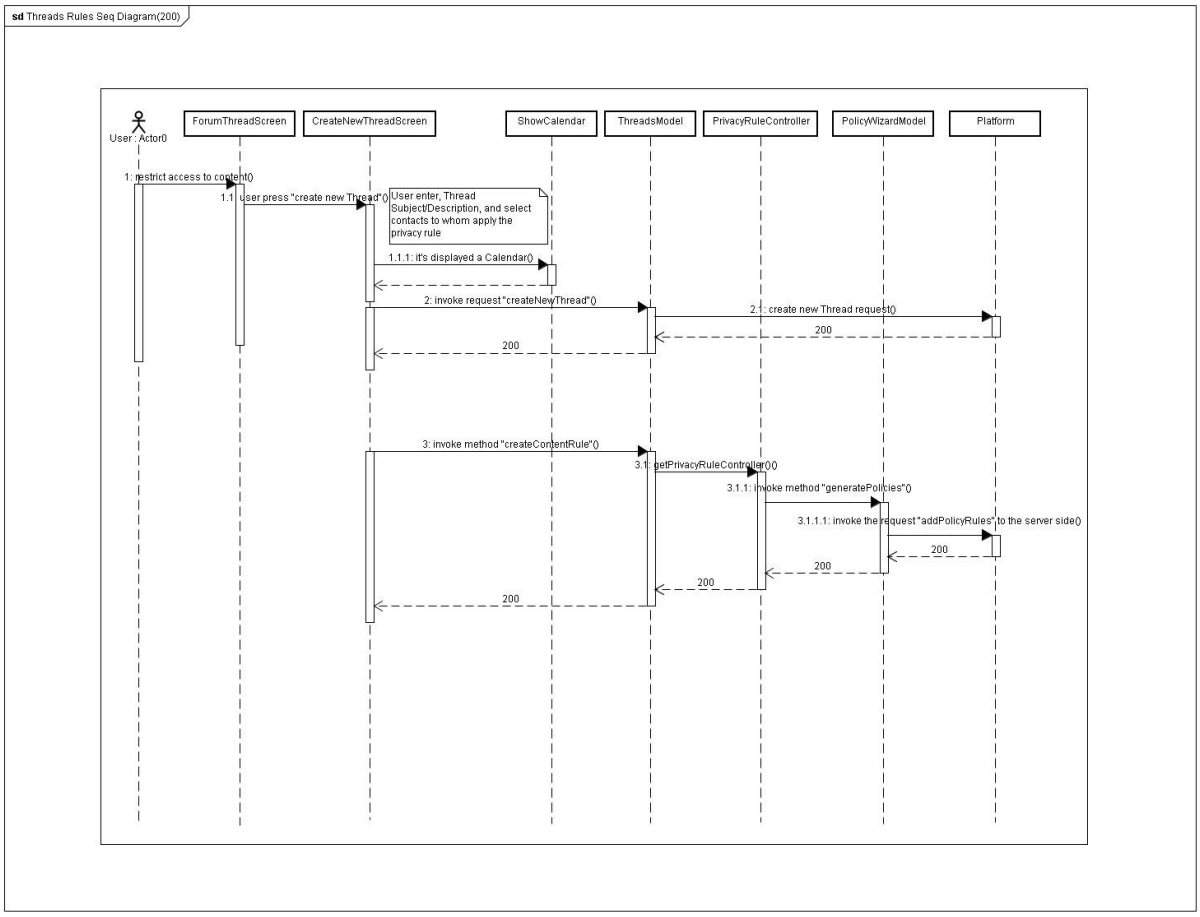


Figure 49: Access restriction to thread content (Posts) sequence diagram

- After successful login in the Gamers application, the user clicks on the "community" tab in the navigation bar and selects the option "Public Forum". As a result, all public community fora are displayed to the user and so he can select one in order to check the available threads.
- In the list of threads, the user presses "Create New Thread" to add a new thread to a forum. A user screen is therefore displayed so that the end user can enter the subject and description for the thread and also set privacy rules.
- To set the privacy rule to the thread, the class **"`CreateNewThreadScreen`"** needs to invoke the class **"`ShowCalendar`"** to display a Calendar where the end-user is able to select a starting/finishing date in which the privacy rule is valid.
- When the user presses the **"`Add thread`"** button two requests will be performed:
- **"`createNewThread`"** is a request to the platform side. If this task is successful (return value is 200) then the new thread is created correctly.
- **"`createContentRule`"** request: if the previous request (`createNewThread`) was successful this request is sent to the server side in order to attach the privacy rule to the thread.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

4.6.1.1 Display and Modify Thread Privacy Rules

After the creation of a new thread subject to a privacy rule, the end user could be interested in modifying this rule to allow/disable contacts to get access to the content (post). The following picture shows the class diagram associated to this use case.

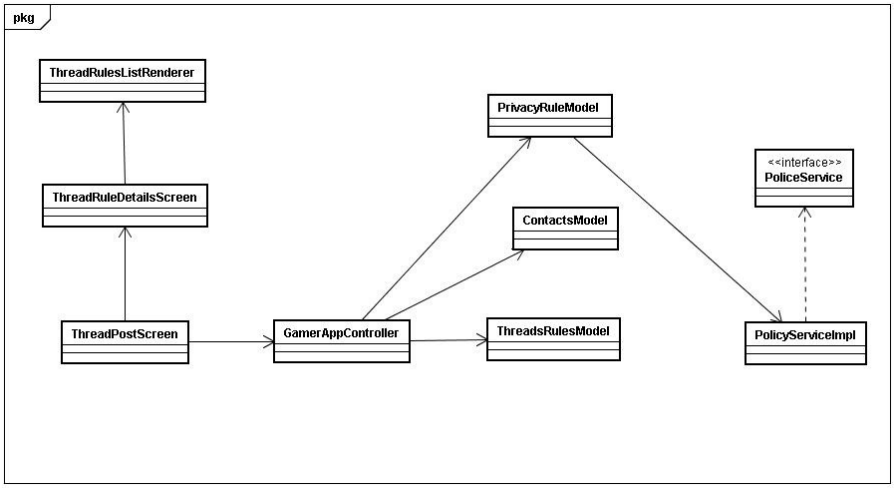


Figure 50: Display, modify and delete thread rules class diagram

These are the involved classes:

- "`ThreadPostScreen`": This screen invokes "`ThreadRuleDetailsScreen`" to display of all posts within a selected thread.
- "`ThreadRuleDetailsScreen`": this is the main screen where the rule details, such as: target contacts, rule type (allow, disallow…) and the valid period of time ("From" and "To" dates), are shown. The screen displays a rule list and two buttons ("change rule" and "delete rule"). In case the user selects "change rule", a container with contacts and the calendar is shown in order to enter again the new rule data. Otherwise (the user selects "delete rule") the selected rule will be removed.
- "`ThreadRulesListRenderer`": this is the renderer class for displaying the rule list.
- "`GamerAppController`": it is the controller class in charge of triggering the screens (displaying the screens to the end-user). It also invokes the class "`model`", in which the request/methods to the platform side are implemented.
- "`ThreadsRulesModel`": This is the class that retrieves the thread rules. Furthermore, it is in charge of retrieving the rule details.
- "`ContactModel`": this class retrieves the contacts from the user's contact list.
- "`PrivacyRuleModel`": this is the model class where the methods "`modifyRule`" and "`deletePolicy`" are implemented. These methods invoke the same methods in "`PolicyServiceImpl`".

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

- "`PolciyServiceImpl`": this class implements the "`PolicyService`" interface that is in charge of performing the requests to the platform side.
- "PoliciyService": this is the interface that implements the requests: "queryPolicy", "setPolicy", "deletePolicy", "modifyPolicyRules".

The figure below shows the sequence diagram for this use case:

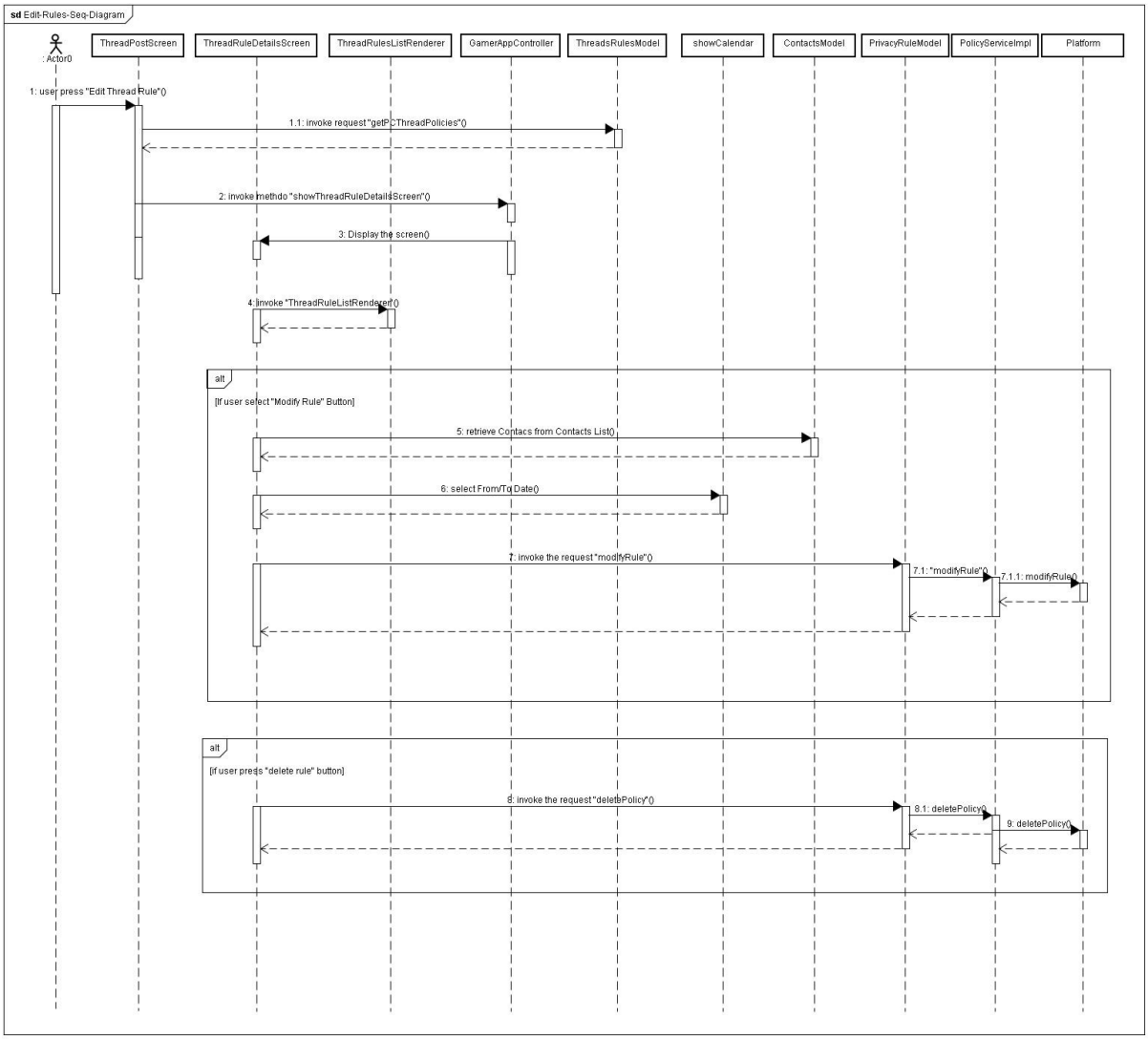


Figure 51: Display, modify and delete thread rules sequence diagram

1. The user selects "`Edit Thread Rule`" from the posts screen, this is, the screen where all posts within that specific thread are displayed.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

2. From "`ThreadPostScreen`" the request "`getPCThreadPolicies`" is invoked in order to retrieve the rules attached to that thread from the platform.
3. From "`ThreadPostScreen`" the GamerAppController request "`showThreadRuleDetailsScreen`" is invoked in order to trigger the screen in charge of displaying the thread rules.
4. The "`ThreadRuleDetailsScreen`" invokes the class "`ThreadRuleListRenderer`" in order to build the rule list.
5. If the end-user selects the "`Modify Rule`" button, then the user's contacts are retrieved from the contacts list, so that the user can enter new contacts associated to that rule.
6. The calendar class is shown again, so that the user can enter a new valid period of time for that rule; this is, "From" date and "To" date.
7. Once the new values are set, the user must press the "`modify`" button, then the request "`modifyPolicy`" will be sent to the PICOS platform.
8. If user selects the "`delete Rule`" button, then the request "`deletePolicy`" will be sent to the PICOS Platform

4.6.2 Content access history

This is a new functionality that is related to requirement "REQ: Gamers R7: Content access history" and is applied to the public community. The following figure shows the class diagram for this feature.

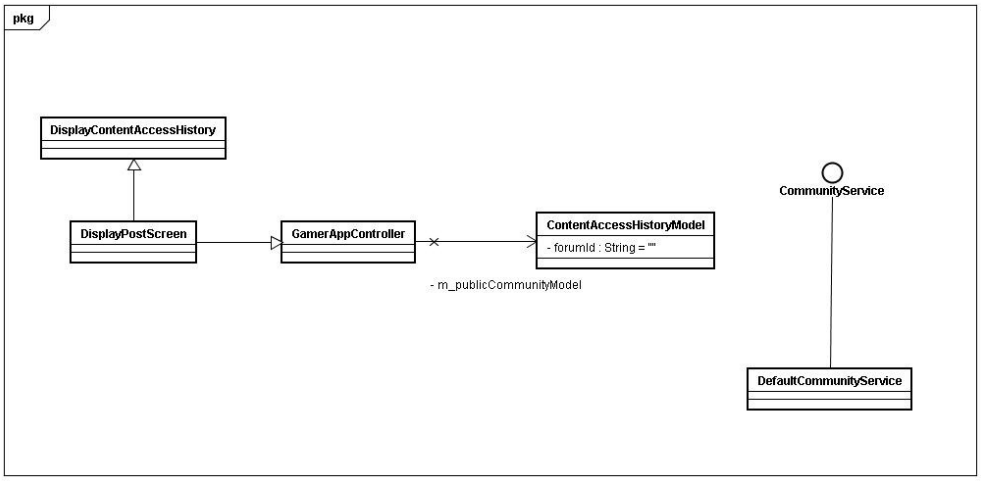


Figure 52: Content access history class diagram

- `DisplayPostScreen:` this class shows the details of the post, such as: post content (text), post owner, a rating list (rates provided by other users) and it also gives the option of performing a rating of the post.
- `DisplayContentAccessHistory:` this is the main screen that displays the access history, showing the pseudonyms of those who got access to the post and the access date.
- `GamerAppController:` this is the controller in charge of launching the screens and triggering the methods of the class "`Model`" (in this case "`ContentAccessHistoryModel`").

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

- `ContentAccessHistoryModel:` this class is in charge of retrieving the history list from the server side and managing the associated actions (add component, delete…).
- `DefaultCommunityService:` In this class the request to the platform "`getContentAccessHistory`" is implemented

Below the picture showing the sequence diagram

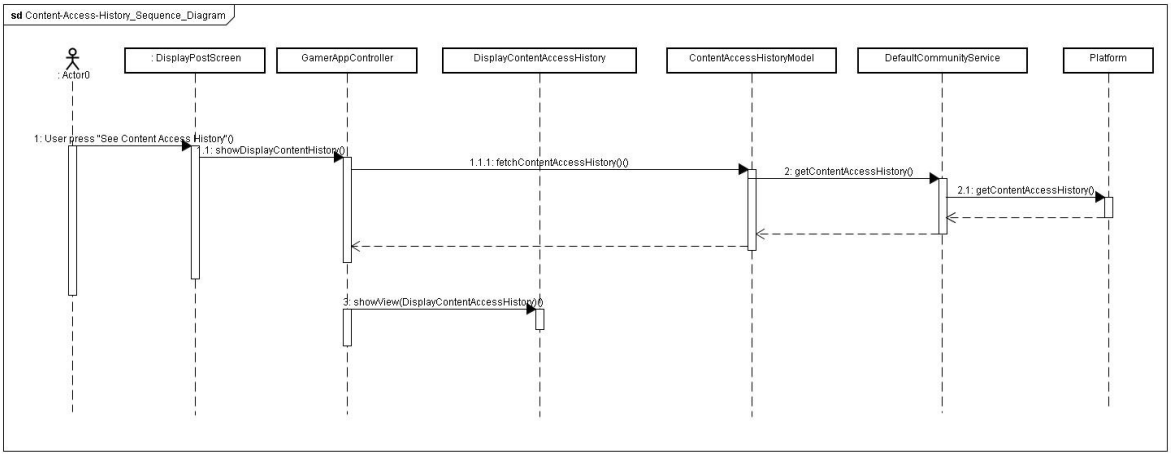


Figure 53: content access history sequence diagram.

4.6.3 Subscription to receiving new available content in threads of the public community.

This new functionality is related to the requirement "REQ: Gamers R5: Notification for new available content". This feature allows the end-user to subscribe to a notification mechanism of receiving an announcement from the server side when a new content/Post is published in a thread within a specific forum of the public community.

See below the class diagram for this use case:

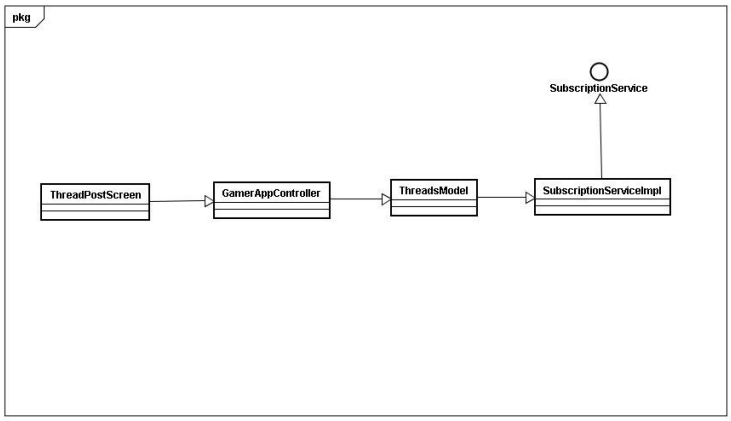


Figure 54: Notification of new available content in forum thread class diagram.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

- `ThreadPostScreen`: this screen displays two options related to a specific thread: "`Subscribe to New Content`" and "`Unsubscribe from New Content`". When the user selects the first option the request "subscribe" to the PICOS Platform is performed, and if the request answer is 0K then the Platform will send a notification to the user informing about new content/Post published in this thread.
- `GamerAppController`: this is the controller in charge of launching the screens and triggering the methods of the class "`Model`" (in this case "`ThreadsModel`")
- `ThreadsModel`: this class implements the methods "`subscribeNewThreadContent`" and "`unsubscribeNewThreadContent`", which invoke the methods "`subscribe`" and "`unsubscribe`" of the "`SubscriptionServiceImpl`"
- `SubscriptionServiceImpl`: this is the class that implements the request methods "`subscribe`" and "`unsubscribe`" to the PICOS platform.

The figure below shows the corresponding sequence diagram:

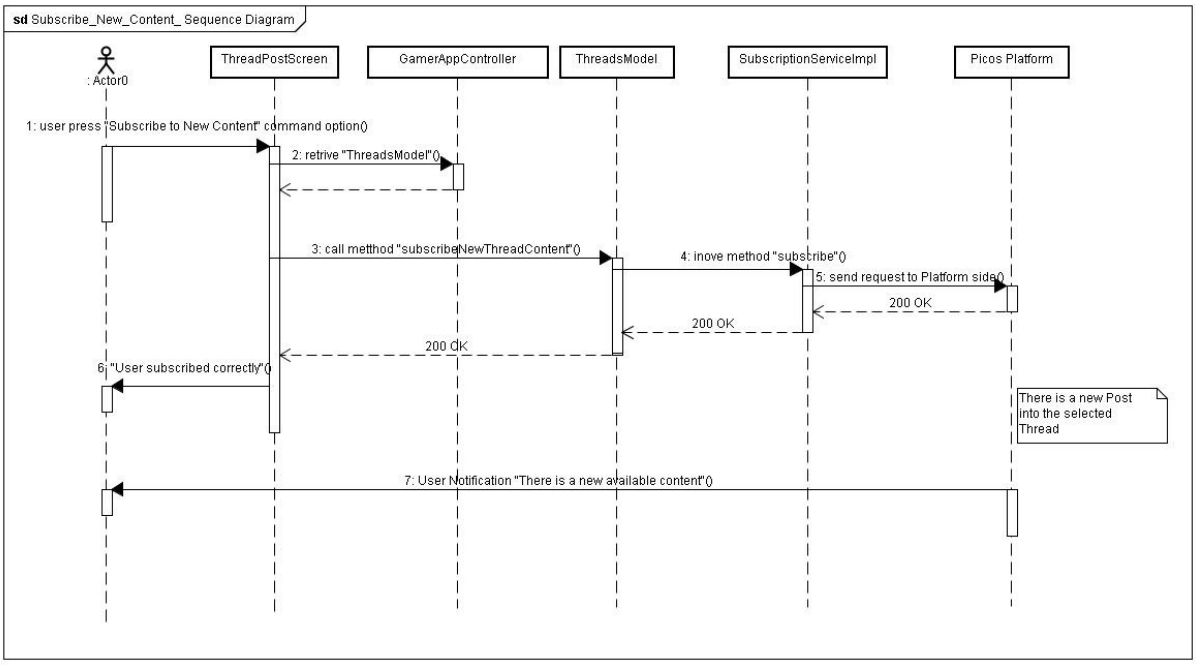


Figure 55: Notification of new available content in forum thread sequence diagram.

1. The user presses "Subscribe to New Content" on the screen where all posts within a thread are displayed.
2. The threads model is retrieved.
3. After retrieval the method "`subscribeNewThreadContent`" is invoked.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

4. The "`subscribeNewThreadContent`" call the method "`subscribe`" from "`SubscriptionServiceImpl`".
5. The request for subscribing to the end-user is sent to the PICOS Platform.
6. The user is informed if the subscription operation was successful
7. The user will receive a notification from Platform when a new content/Post is available into the thread.

4.7 Public Repository

4.7.1 Access restriction to published content in the Public Repository (Categories)

This functionality is related to requirements "REQ: Gamers R3: restrict access to published content" and "REQ: Gamers R4: restrict access based on date condition" applied to categories in the public repository. The main involved classes in this scenario are the following:

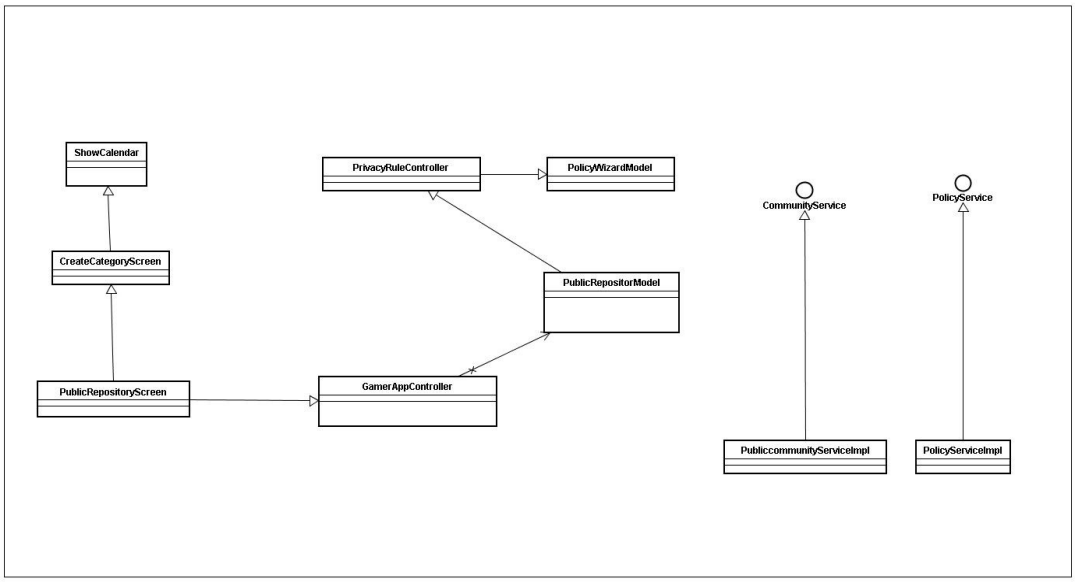


Figure 56: Access restriction to content in the Public Repository (Categories) class diagram.

- "`PublicRepositoryScreen`": it is in charge of retrieving and displaying the category list from the platform.
- "`CreateCategoryScreen`": this class displays a screen where the user is able to enter a name, description and key words for the category. Furthermore, the user is allowed to attach a privacy rule to this category, setting those contacts that will have access to the content within (file, pictures.) and the valid period of time for this rule.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

- "`ShowCalendar`": this displays the calendar, where the end-user can select the starting/finishing time for the privacy rule.
- "`GamerAppController`": is the controller class in charge of triggering the screens (displaying the screens to the end-user). It also invokes the class "`model`" in which the request/methods to the platform side are implemented.
- "`PrivacyRuleController`": is the controller in charge of invoking the "`PolicyWizardModel`", containing the methods for adding and removing privacy rules.
- "`PublicRepositoryModel`": this is the model class that implements those methods for retrieving the category content, adding new content to the repository, removing a selected content and attaching a privacy rule to the category itself or to a selected content.
- "`PubliccommunityServiceImpl`": this is the class where the requests to the platform for managing the repository categories are implemented: "retrieve the category list", "add/remove/update category", "upload/remove content into the repository category".
- "`PolicyServiceImpl`": this class implements the requests to the platform side with regard to policies: "add", "remove", "modify" and "evaluate".

Figure below shows the sequence diagram for this feature.

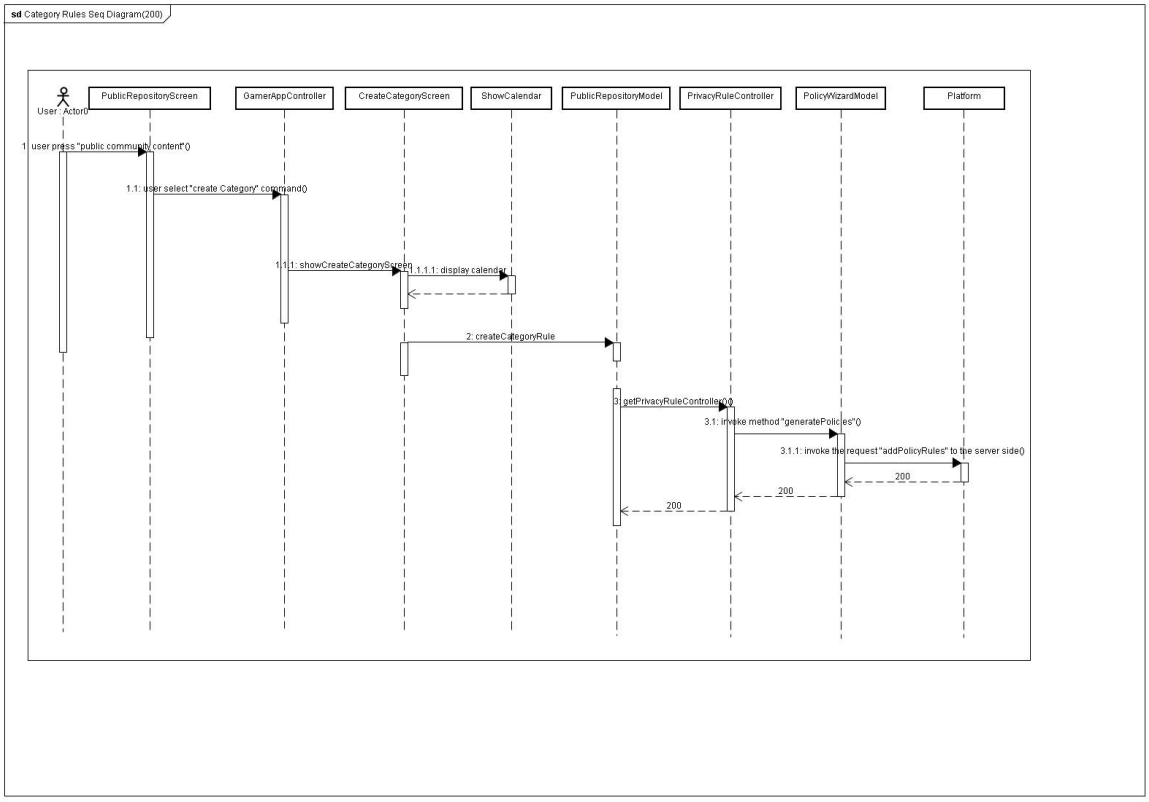


Figure 57: Access restriction to content in the Public Repository (Categories) sequence diagram.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

4.7.2 Access Restriction to Category Files

This functionality is related to requirements "REQ: Gamers R3: restrict access to published content" and "REQ: Gamers R4: restrict access based on date condition" applied to files within categories in the public repository.

Next figure shows the classes involved in this use case:

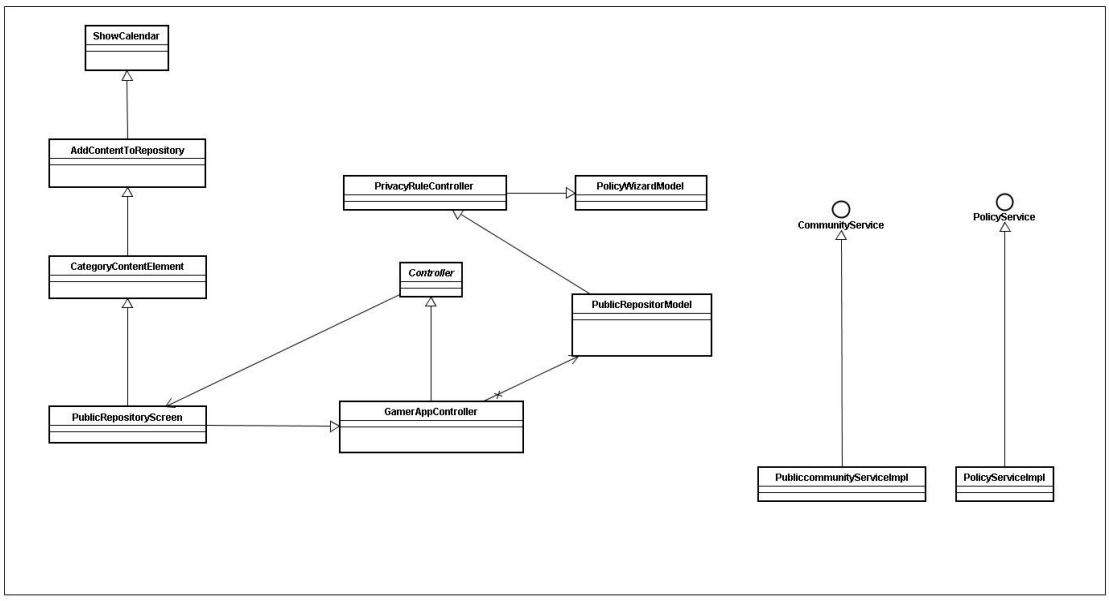


Figure 58: Access restriction to content in the Public Repository class diagram.

The main involved classes in this scenario are the following:

- "`PublicRepositoryScreen`": is in charge of retrieving and displaying the category list from the platform.
- "`CategoryContentElement`": is in charge of retrieving and displaying the content (files, pictures...) for a selected category. It also allows creating a new category and rating content.
- "`AddContentToRepository`": it displays the screen where the end-user is able to upload new content to the repository category. During the step of uploading new content, the user is able to attach a privacy rule to this content (file, picture...). This class invokes the class "`ShowCalendar`" to display a calendar where end-user can set a valid period time for the privacy rule.
- "`ShowCalendar`": this displays the calendar, where the end-user can select the starting/finishing time for the privacy rule.
- "`GamerAppController`": is the controller class in charge of triggering the screens (displaying the screens to the end-user). It also invokes the class "`model`" in which the request/methods to the platform side are implemented.

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

- "`PrivacyRuleController`": is the controller in charge of invoking the "`PolicyWizardModel`", which contains the methods for adding and removing privacy rules.
- "`PublicRepositoryModel`": this is the model class that implements those methods for retrieving the category content, adding new content to the repository, removing a selected content and attaching a privacy rule to the category itself or to a selected content.
- "`PubliccommunityServiceImpl`": this is the class where the requests to the platform for managing the repository categories are implemented: "retrieve the category list", "add/remove/update category", "upload/remove content into the repository category".
- "`PolicyServiceImpl`": this class implements the requests to the platform side with regard to policies: "add", "remove", "modify" and "evaluate".

Figure below shows the sequence diagram for this feature.

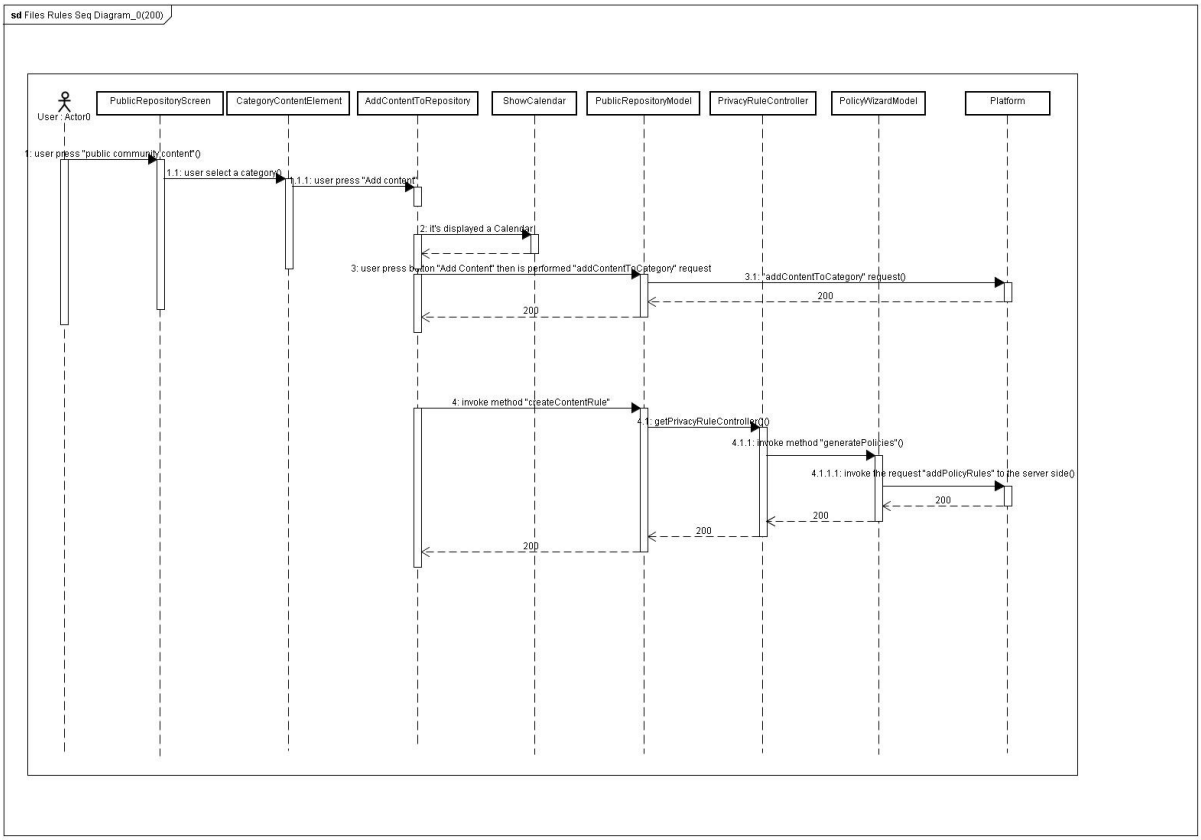


Figure 59: Access restriction to content in the Public Repository sequence diagram.

4.8 My Files

This functionality is a merge of requirements "REQ : Gamers R9: Shared desk " and "REQ: Gamers R12: access to private room".

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

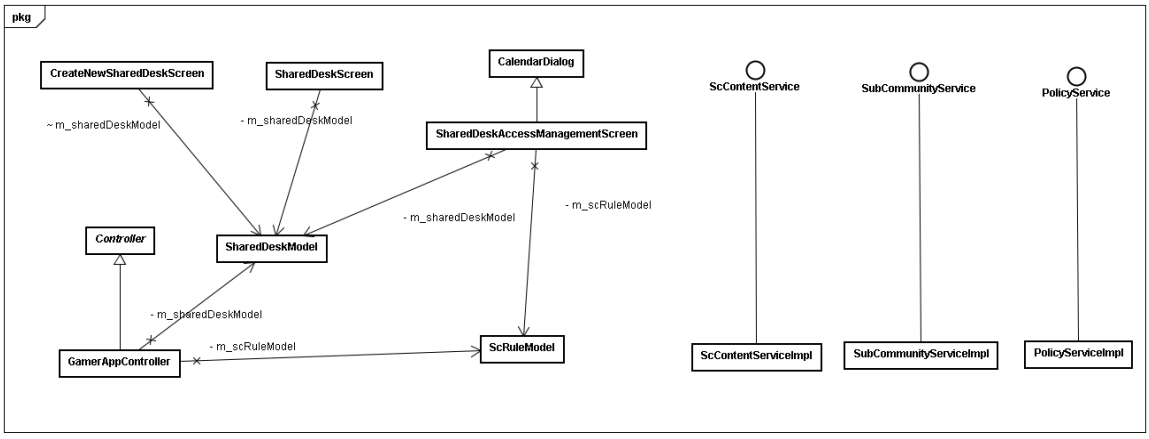


Figure 60: Shared Desk class diagram

The main involved classes in this scenario are the following:

- "`CreateNewDeskScreen`": this class implements the screen to create a new shared Desk.
- "`SharedDeskAccessManagementScreen`": this class implements the screen to invite contacts to a shared Desk and remove contacts from a shared Desk.
- "`SharedDeskScreen`": this class implements the screen to display the shared desk. It displays the content list of a shared Desk.
- "`SharedDeskModel`": this model is in charge of managing the shared Desk and the shared Desk content for a user.
- "`ScRuleModel`": implements the methods for adding and removing privacy rules for the access to a shared Desk.
- "`CalendarDialog`": this displays the Calendar, where end-user can select the starting/finishing time for the privacy rule.
- "`GamerAppController`": it is the controller class in charge of triggering the screens (displaying the screens to the end-user). It also invokes the "model" class in which the request/methods to the platform side are implemented.
- "`PolicyServiceImpl`": this class implements the requests to the platform side with regard to policies: "add", "remove", "modify" and "evaluate".
- "`ScContentServiceImpl`": this is the class where the requests to the platform for managing the Shared Desk content are implemented, such as: "create and remove content" or "retrieve the content list".
- "`SubCommunityServiceImpl`": this is the class where the requests to the platform are implemented to get the shared Desk list, create a new shared Desk or close/leave a shared Desk.

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

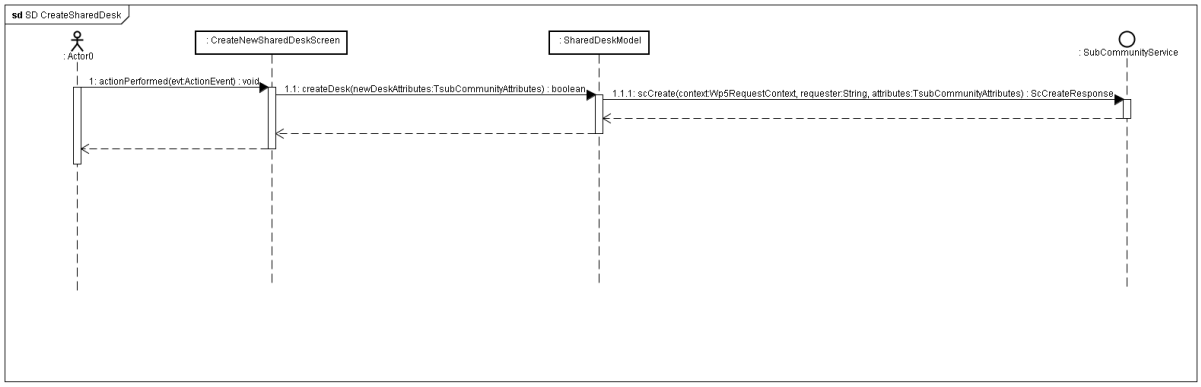


Figure 61: Create new shared Desk sequence diagram

After the user filled in the title and description of the shared Desk and pressed the "create new shared Desk" button. The "`createDesk`" method invokes the platform request method "`scCreate`" in the `SharedDeskModel`.

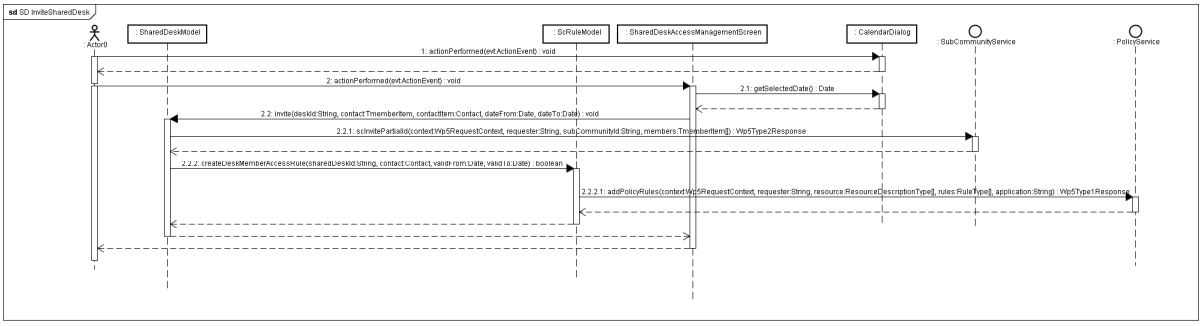


Figure 62: Invite contact to a shared Desk sequence diagram

1. The user selects in the "CalendarDialog" a start and end date where the contacts have access.
2. If the User pressed the invite button the "`SharedDeskAccessManagementScreen`" gets the selected dates from the "`CalendarDialog`". The invite method invokes the "`scInvite`" and "`createDeskMemberAccessRule`" method in the `SharedDeskModel`.
   - The "`scInvite`" method is a server request to invite a contact to join the shared Desk.
   - The "`createDeskMemberAccessRule`" method creates a policy rule, using the server request method "`addPolicyRules`" in the "`ScRuleModel`", which allows the contact to view the shared Desk content.

4.9 Sub-Communities

4.9.1 Access restriction to published content in Sub-Communities

This functionality is related to requirements "REQ: Gamers R3: restrict access to published content" applied to sub-communities.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

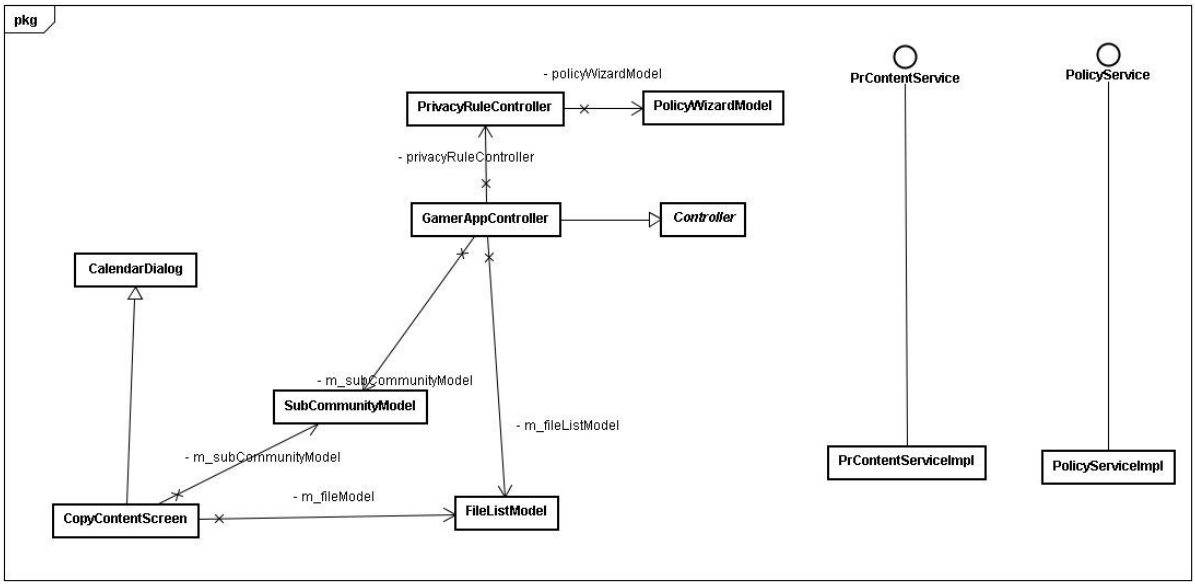


Figure 63: Class diagram for access restriction to published content in the Sub-Communities.

The main involved classes in this scenario are the following:

- "`CopyContentScreen`": this class implements the screen to add a selected content to a Sub-Community. The end-user will be able to select a Sub-Community and also attach a privacy rule.
- There are three possible options:
  1. Everybody (the rule applies to all members of the selected Sub-Community)
  2. All contacts from the user's contact list which are a member in the Sub-Community
  3. Selected contact's from the Sub-Community member list
- The class also invokes the "`ShowCalendar`" class in order to display a calendar, allowing the user to specify a starting/finishing date for the rule.
- "`SubCommunityModel`": this is the model which is in charge of managing the Sub-Communities for a user. It retrieves the Sub-Community list from the server side.
- "`FileListModel`": this is the model which is in charge of managing the content of a user. It retrieves the content list from the server side and implements the methods for adding content to a Sub-Community.
- "`ShowCalendar`": this displays the Calendar, where end-user can select the starting/finishing time for the privacy rule.
- "`GamerAppController`": this is the controller class which is in charge of triggering the screens (displaying the screens to the end-user). It also invokes the "model" class in which the request/methods to the platform side are implemented.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

- "`PrivacyRuleController`": it is the controller which is in charge of invoking the "`PolicyWizardModel`", which implements the methods for adding and removing privacy rules.
- "`PolicyServiceImpl`": this class implements the requests to the platform side with regard to policies: "add", "remove", "modify" and "evaluate".
- "`PrContentServiceImpl`": this is the class where the requests to the platform for managing the Pr-content are implemented, such as: "create/remove/update content", "retrieve the content list", "copy content to a Sub-Community".

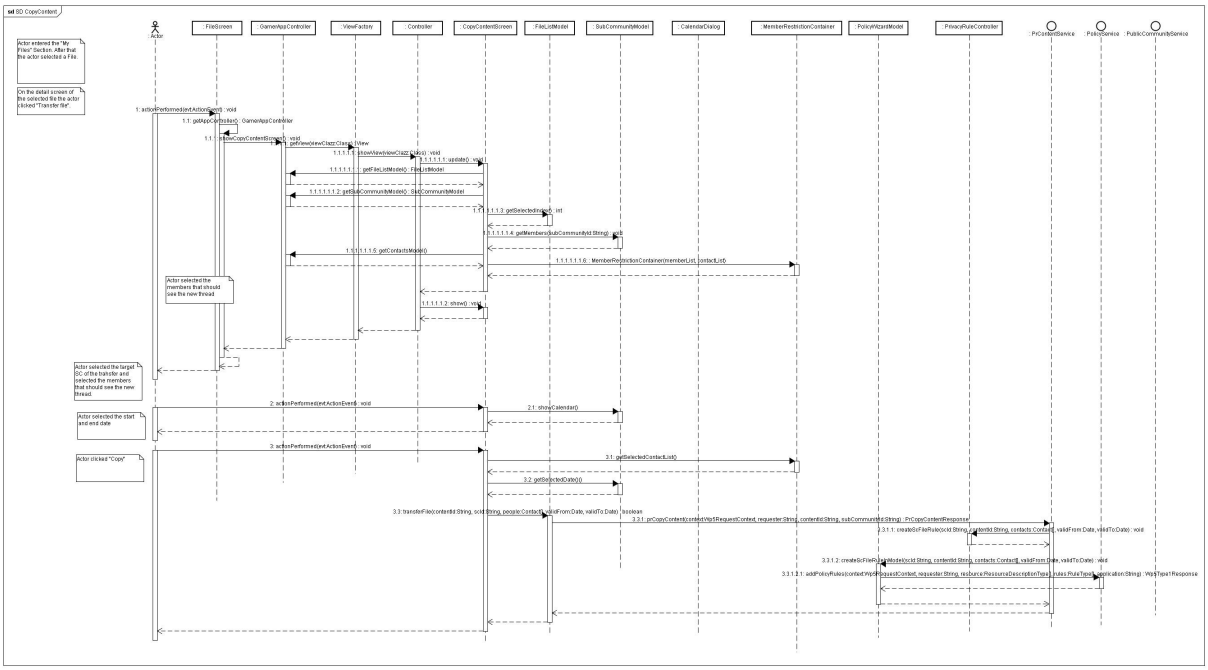


Figure 64: Rules attached to published content Sequence Diagram

1. The User clicks the transfer button to open the "`CopyContentScreen`" to copy the file to a SubCommunity. The "`CopyContentScreen`" receive the SC-memberlist and the contactlist is displayed to the "`MemberRestrcitionContainer`" Container.
2. The user presses the calendar button to select the start and end date for the privacy rule
3. When the user presses the "`Copy Content`" button:
   - 3.1 the "`CopyContentScreen`" class gets the selected members from the "`MemberRestrcitionContainer`"
   - 3.2 the "`CopyContentScreen`" class gets the selected start and end date from the "`CalendarDialog`"
   - 3.3 "`prCopyContent`" is a request to the platform side. If this task is successful (return value is 200) the new thread is created correctly.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

3.4 "`addPolicyRules`" request: if the previous request (`prCopyContent`) was successful this request is sent to the server side in order to attach the privacy rule to the thread.

4.9.2 Access Restrictions based on date conditions

This functionality is related to requirement "REQ: Gamers R4: restrict access based on date condition" applied to sub-communities. The main involved classes in this scenario are the following:

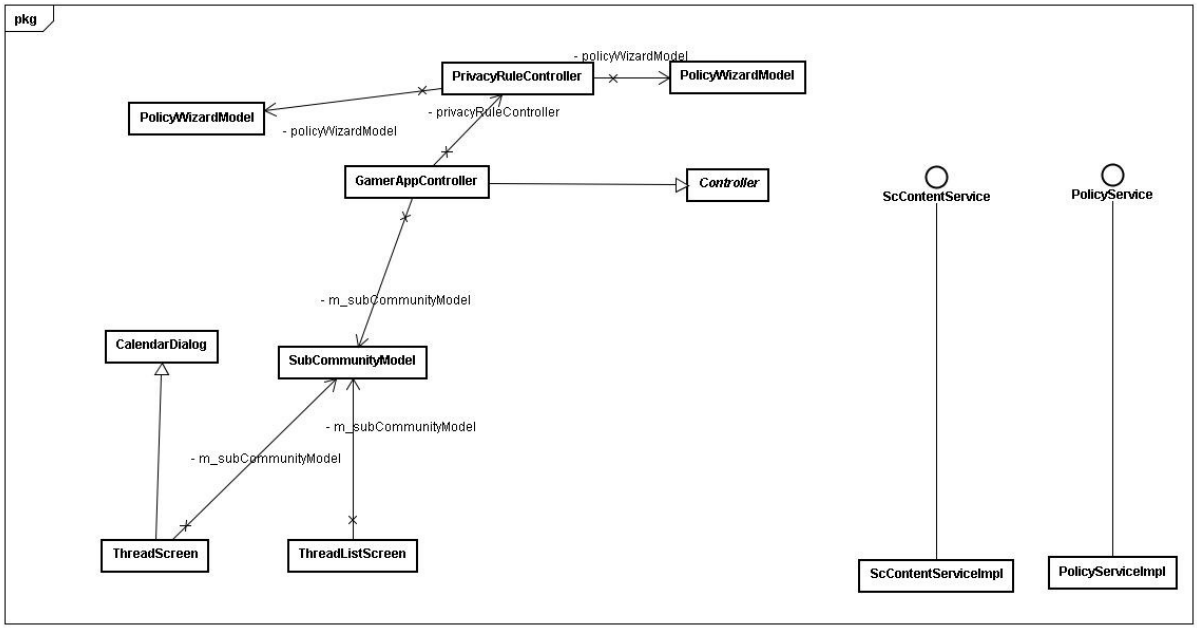


Figure 65: Class diagram for access restriction to based on date conditions in the Sub-Communities.

- `ThreadScreen`: this class implements the screen to add a thread to a Sub-Community. The end-user will be able to enter a subject and also to attach a privacy rule.
- There are three possible options:
  1. Everybody (the rule applies to all members of the selected Sub-Community)
  2. All contacts from the user's contact list which are member in the Sub-Community
  3. Selected contact's from the Sub-Community member list
- The class also invokes the "`ShowCalendar`" class in order to display a calendar, allowing the user to specify a starting/finishing date for the rule.
- "`SubCommunityModel`": this is the model which is in charge of managing the Sub-Communities for a user. It retrieves the Sub-Community list from the server side.
- "`ShowCalendar`": this displays the Calendar, where end-user can select the starting/finishing time for the privacy rule.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

- "`GamerAppController`": this is the controller class which is in charge of triggering the screens (displaying the screens to the end-user). It also invokes the "model" class in which the request/methods to the platform side are implemented.
- "`PrivacyRuleController`": this is the controller which is in charge of invoking the "`PolicyWizardModel`", which implements the methods for adding and removing privacy rules.
- "`PolicyServiceImpl`": this class implements the requests to the platform side with regard to policies: "add", "remove", "modify" and "evaluate".
- "`ScContentServiceImpl`": this is the class where the requests to the platform for managing the Sc-content are implemented, such as: "create/remove/update content", "retrieve the content list", "copy content to a Sub-Community".

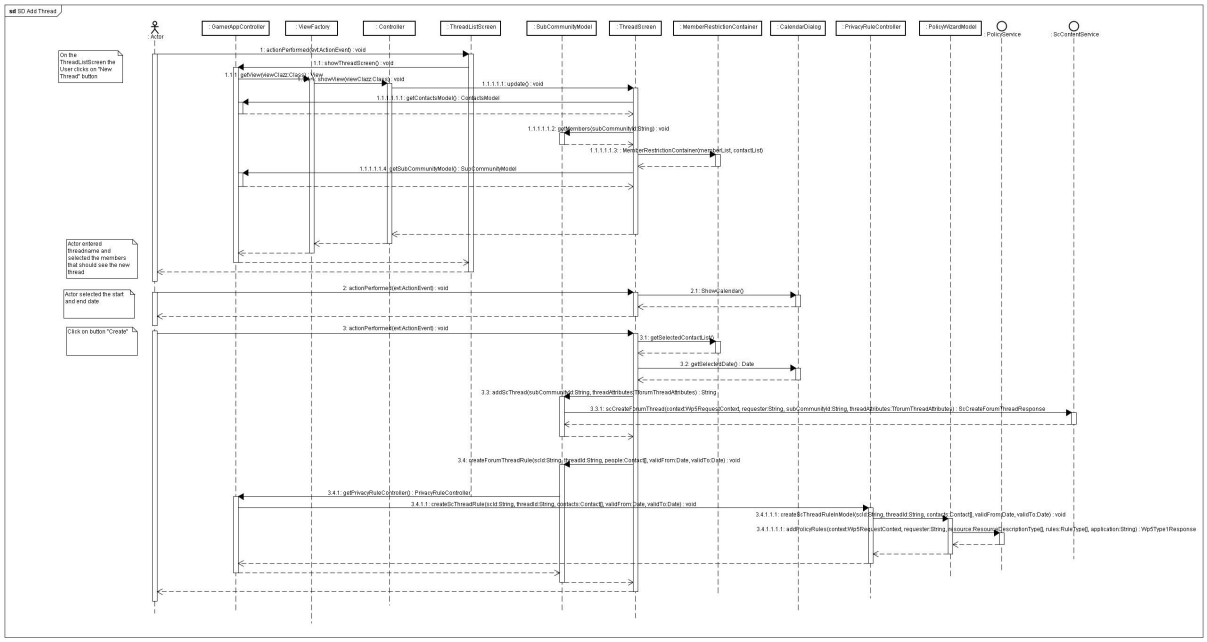


Figure 66: Rules attached to Sub-Community content Sequence Diagram

1. On the mobile screen the user presses "`New Thread`" to add a new thread into a Sub-Community forum. A user screen is therefore displayed so that the user can enter the subject of the thread and set privacy rules which will be attached to it. The "`ThreadScreen`" receives the SC-memberlist and the contact list to display the "`MemberRestrcitionContainer`" Container.
2. The user presses the calendar button to select the start and end date for the privacy rule
3. When the user presses the "`Add thread`" button:

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

3.1 the "`ThreadScreen`" class gets the selected members from the "`MemberRestrcitionContainer`"

3.2 "`addScThread`" is a request to the platform side. If this task is successful (return value is 200) then the new thread is created correctly.

3.3 "`createForumThreadRule`" request: if the previous request (`addScThread`) was successful then this request is sent to the server side in order to attach the privacy rule to the thread.

4.10 Location Based services

4.10.1 Point Of Interest

This chapter addresses the following requirements for the gamer prototype:

- REQ: Gamers R16: public Point of interest
- REQ: Gamers R19: custom POI
- REQ: Gamers R20: private or public POI

A Point Of Interest (POI) in the context of the picos gamer prototype is described by title, description, image, coordinates and visibility (public/private).

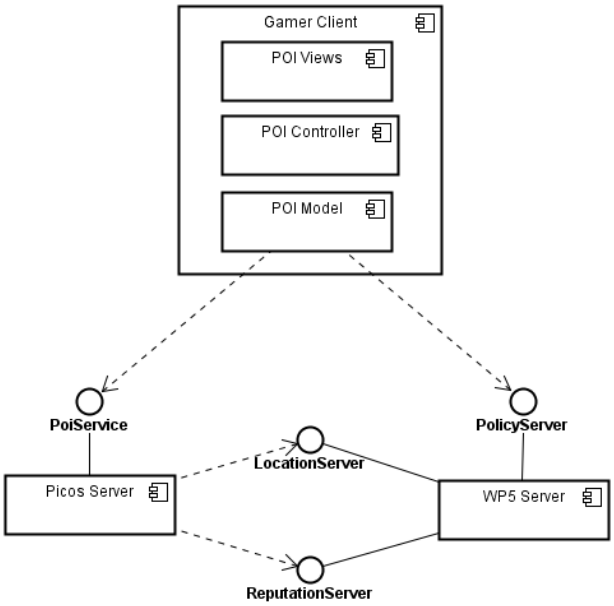


Figure 67: POI deployment diagram.

The gamer client is separated into the typical Model-View-Controller (MVC) layers. The controller mediates between the view and the model layer and controls the sequence of views and the handling of

Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

exceptions. The views display data retrieved from the model and forward data retrieved from the user to the controller. The model holds the state of the POI list and interacts with the server components.

The PICOS server aggregates data retrieved from the wp5 server and caches/resizes images.

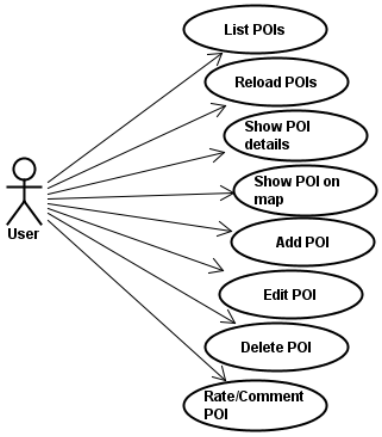


Figure 68: POI usecase diagram.

List POIs

Starting from the "Map" tab the user can list all POIs that are visible to him depending on the POI visibility. The thumbnail image generation and the aggregation of reputation data is performed by the PICOS server.

Reload POIs

For performance reasons the POI list is persistently cached on the client. An update is only enforced before a POI is edited or deleted. When displaying the POI list for the first time after a login the user is asked if he wants to reload the POI list. The list may be reloaded manually at a later point in time.

Show POI details

The POI details can be displayed by clicking the POI's icon on the map or by selecting the POI in the POI list. The displayed data are taken from the cached POI list. When the thumbnail image is clicked the zoom image is retrieved from the PICOS server.

Show POI on map

The POIs can be displayed on the map. When you select a POI from the POI list or start from the "Show POI details" view, the map will be centered on the POI's location. All POIs that fit on the current map section will be displayed by POI icons.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Add POI

Starting from the Map or from the POI list a user can add a new POI by entering title and description, capturing or opening an image and entering or selecting the coordinates. When selecting the coordinates on the map, the map will initially be centered on the previously entered POI coordinates or - if not available - on the current user location.

The visibility of a POI can either be public (visible to every member of the public community) or private (visible only to the owner).

Edit POI

Only the owner of a POI is authorized to edit the POI. The latest version of the POI is reloaded from the PICOS server to make sure that no stale version from the local cache is used. Title, description, image, coordinates and visibility can be edited.

Delete POI

Only the owner of a POI is authorized to delete the POI. The latest version of the POI is reloaded from the PICOS server to make sure that the POI still exists.

Rate/Comment POI

A POI can be rated/commented by users (except by the owner of the POI). The average rating and the rating history of a POI are aggregated into the POI data by the PICOS server.

4.10.2 Point Of Interest Recommendation

A POI can be recommended from a user to one or more of his contacts. The user selects a POI from the POI list, selects the users that will receive the recommendation from a list of his contacts and enters a recommendation message.

Users have the chance to subscribe or unsubscribe for POI recommendations on the user's settings screen. They receive asynchronous recommendations only if they have subscribed to the POI recommendation service.

The recommendation notification displays the recommendation message and a link which forwards to the "Show POI details" view of the recommended POI when clicked.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

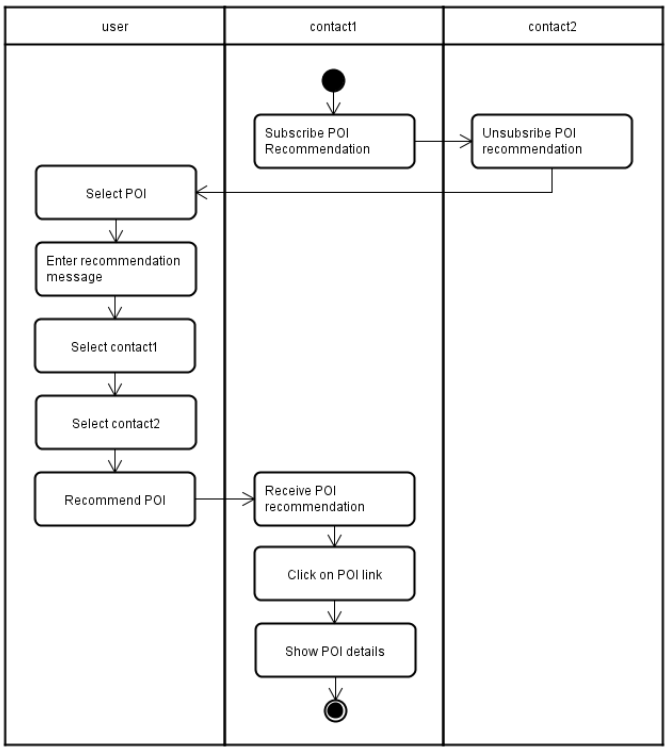


Figure 69: POI recommendation activity diagram.

4.10.3 Private Site

Private Sites are areas that have special meaning to their owner, e.g. home or work. They consist of title, description, location and size and are only directly visible to their owner. A Private Site can be referenced in a policy which means that the policy will be evaluated only if the owner's current location is inside the referenced Private Site.

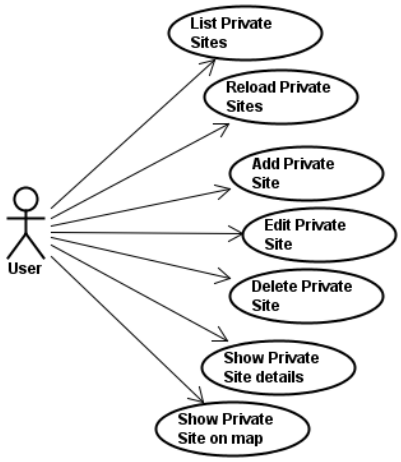


Figure 70: Private Site usecase diagram.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

List Private Sites

Starting from the "Map" tab the user can list all his Private Sites.

Reload Private Sites

For performance reasons the Private Site list is persistently cached on the client. An update is only enforced before a Private Site is edited or deleted. When the user lists Private Sites for the first time after a login he is asked if he wants to reload the Private Site list. The list may be reloaded manually at a later point in time.

Show Private Site details

The Private Site details can be displayed by clicking the Private Site's icon on the map or by selecting the Private Site in the Private Site list. The displayed data are taken from the cached Private Site list.

Show Private Site on map

The Private Sites can be displayed on the map. When you select a Private Site from the Private Site list or start from the "Show Private Site details" view, the map will be centered on the Private Site's location. All Private Sites that fit on the current map section will be displayed by Private Site icons.

Add Private Site

Starting from the Map or from the Private Site list a user can add a new Private Site by entering title and description, entering or selecting the coordinates and selecting the size. When selecting the coordinates on the map, the map will be initially centered on the previously entered coordinates of the Private Site or - if not available - on the current user location.

Edit Private Site

The latest version of the Private Site is reloaded from the orchestration layer to make sure that no stale version from the local cache is used. Title, description, coordinates and size can be edited.

Delete Private Site

The latest version of the Private Site is reloaded to make sure that the POI still exists.

4.10.4 List Nearby Users

This section addresses the requirement “REQ : Gamers R17: meeting with nearby players”

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community’s Seventh Framework Programme.

Starting from the "Map" tab a user opens the "List Nearby Users" view. Initially the list is empty and the user selects the size of the area around his current location. Then the user receives a list of users whose current location is inside the selected area and whose policy and location sensor settings allow to locate them. The user may select one or more of the nearby users and initiate an action (send message, invite to chat, add to contacts, show on map) on the selected set of nearby users.

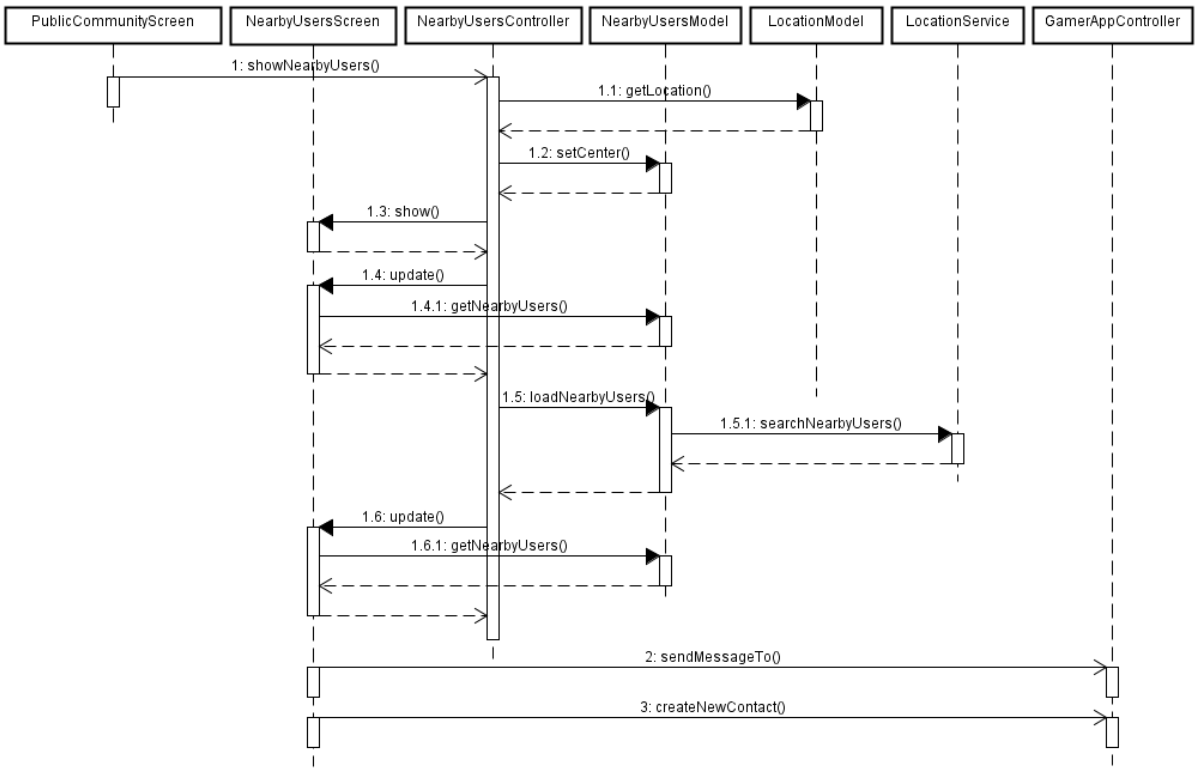


Figure 71: List Nearby Users sequence diagram.

4.11 Real-time Content Sharing: Communication and Notification Means

This refers to “REQ: Gamers R21: Real time content sharing” and “REQ: Gamers R22: archive chat”. No changes with regard to D6.1 (section 2.2.5, page 60) [3]

4.12 Notifications

Components, which want to send the mentioned notifications, now receive a status response “302” when the destination is not logged in and when the notification is queued on a per destination basis.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community’s Seventh Framework Programme.

Some of the notification requires user feedbacks to be processed by the components. As the notification is then asynchronously processed by the client application, the component must offer new entry points to receive the notification feedback.

The following diagram shows the different processing of the notification when the destination is on-line or off-line.

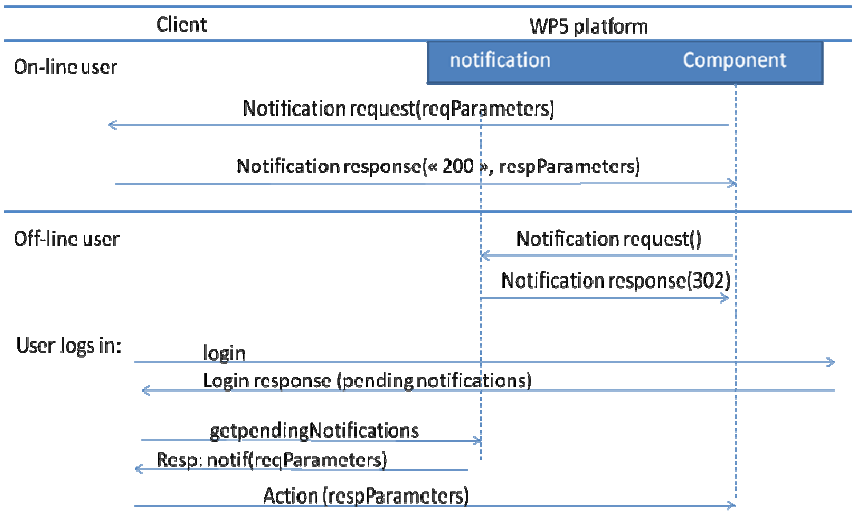
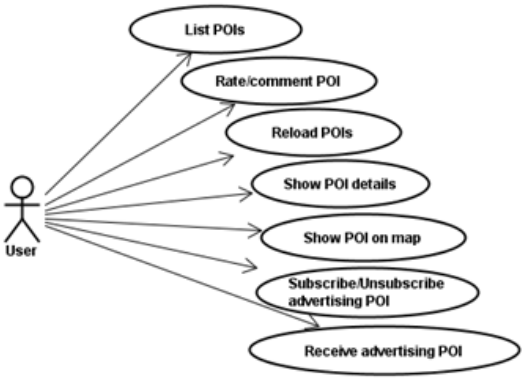


Figure 72: Offline notifications diagram

4.13 Advertising services

Advertising services include the possibility for a commercial user to create a special kind of Point Of Interest (advertising POI) that includes a target user profile. By matching the profile of users against the target user profile the WP5 server platform decides whether the advertising POI is suitable for a user or not.

Advertising services consist of the following use cases:



Copyright © 2008-2010 by the PICOS consortium - All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

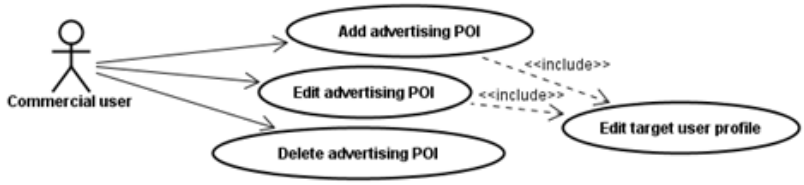


Figure 73: Advertising Services usecase diagram.

In the gamer application prototype users and commercial users (advertisers) are simply distinguished by their display name: display names ending with ".com" belong to commercial users.

Most usecases of the user are identical to the ones with POIs. "List POIs", "Rate/comment POI", "Reload POIs", "Show POI details" and "Show POI on map" do not differentiate between POIs and advertising POIs (See section 4.10.1 for more information about Points of Interest).

The usecases of the commercial user are similar to the ones of the user with the difference, that the commercial user edits the target user profile as part of the "Add advertising POI" and "Edit advertising POI" usecases. In addition advertising POIs are always public (they cannot be declared private).

Subscribe/Unsubscribe advertising POI

Users have the option to subscribe or unsubscribe for advertising POIs. They receive asynchronous notifications when they come close to an advertising POI only if they have subscribed to the advertising POI service and their profile matches the target user profile of the advertising POI. Subscribe/Unsubscribe may be performed in the user's settings screen.

Receive advertising POI

When a user that subscribed to the advertising POI service comes close to an advertising POI and his profile matches the target user profile of the advertising POI the user receives an asynchronous advertising POI notification. The notification states that the user might be interested in the advertising POI and a link which forwards to the "Show POI details" view of the advertising POI when clicked.

Edit target user profile

After a commercial a user clicks the "Publish" button in the "AddPoi" or "EditPoi" Screen and is forwarded to the "EditTargetUserProfile" Screen. The more specific the target user profile is (the more data it contains) the less users will match that profile. The exact profile matching algorithm is implemented in the wp5 server platform and therefore not a topic of this documentation.

4.14 Revocation

This functionality refers to “REQ: Anglers R37: revocation”. No changes on the client side from D6.1 Section 2.2.17 (page 115) [3]. Most changes apply to the server side.

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community’s Seventh Framework Programme.

Appendix A List of R2 implemented requirements in Gamers prototype.

According to priority:

Necessary requirements

- REQ : Gamers R1: Sharing Contact-List
- REQ : Gamers R3: restrict access to published content
- REQ : Gamers R4: restrict access based on date condition
- REQ : Anglers R34: Enhance Content Awareness.
- REQ : Anglers R36: social presence awareness (POI)

Recommended requirements

- REQ: Gamers R5: Notification for new available content
- REQ: Gamers R7: Content access history
- REQ: Gamers R16: public Point of interest
- REQ: Gamers R19: custom POI
- REQ: Gamers R20: private or public POI
- REQ: Gamers R23: Advertising services
- REQ: Anglers R37: revocation

Helpful requirements

- REQ: Gamers R9: Shared desk (merged with
- REQ: Gamers R10: Real chat
- REQ: Gamers R17: meeting with nearby players
- REQ: Gamers R21: Real time content sharing
- REQ: Gamers R22: archive chat
- REQ: Anglers R26: Offline notifications

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.

Bibliography & References

[1] PICOS_Proposal_Replanning_2nd_Cycle_01122009.doc.

[2] R2 Investigation Report (Internal deliverable). PICOS_Investigation_Report _v1_0_Final.doc

[3] Deliverable D6.1 Community Application Prototype: http://picos-project.eu/fileadmin/user_upload/fmgr/Deliverables/WP6_Application_Prototype/D6.1_Community_application_prototype_1/PICOS_D6_1_Community_Application_Prototype_v1_Final_Public.pdf

[4] Deliverable D6.2a Community Application Prototype: http://picos-project.eu/fileadmin/user_upload/fmgr/Deliverables/WP6_Application_Prototype/D6.2_Community_application_prototype_2/PICOS_D6_2_Community_Application_Prototype2_v1_Final.pdf

[5] Deliverable D5.1a Platform Prototype 1: http://picosproject.eu/fileadmin/user_upload/fmgr/Deliverables/WP5_Platform/D5.1_Platform_prototype_1/PICOS_D5_1_Platform_Prototype_1_v1_1_Final_Public.pdf

[6] Deliverable D5.2a Platform prototype 2a: http://www.picos-project.eu/fileadmin/user_upload/fmgr/Deliverables/WP5_Platform/D5.2_anglers_prototype/PICOS_D5_2a_platform_prototype_1_2_final.pdf

[7] Deliverable D4.2 Platform Architecture and Design 2.

[8] Deliverable D7.2b "First Community Prototype: Field Trial Report" http://picos-project.eu/fileadmin/user_upload/fmgr/Deliverables/WP7_Trials/D7.2b_Field_Trial_Report/Picos_D7_2b_Field_Test_Report.pdf

[9] Deliverable D7.1b Trial plan for the second community prototype.

[10] ICT 2010 "Digitally Driven": http://ec.europa.eu/information_society/events/ict/2010/index_en.htm

[11] Anglersbase: http://www.anglersbase.org/

Copyright © 2008-2010 by the PICOS consortium – All rights reserved.

The PICOS project receives research funding from the Community's Seventh Framework Programme.