Bilkent University
Department of Computer Engineering

CS 492
Senior Design Project
Final Report
4/20/2016

GrouPub

- A Location-Based Quiz Application –

Group Members:

Arda Ekmekçi	21101065
Ayberk Aksoy	21100623
Ekin Karayalçın	21101919
Merve Tuncel	21102000
Seren Erdoğan	21100693

Supervisor: Fazlı Can
Jury Members: Selim Aksoy, Hakan Ferhatosmanoğlu
Expert: Mehmet Çakır
Project Website: http://groupub.github.io/

Table of Contents

List of Figures

1. Abstract

GrouPub is a location-based quiz application where users compete against each other in quiz events that are hosted in specific pubs or cafes. The need of an application like GrouPub starts with how dull one's life can become because of the work life/daily responsibilities and need to find a good and fun way to spend his/her time with friends or new people he/she meets via the application. The user can download the application from store and can easily figures how the application works. The user creates a group or apply to an existing one and the address of the café is provided to user. When the quiz starts the user and the users group participates in the event together. The winner team gets free drinks/rewards from the café even tough in the end everyone wins since everyone spends quality time with their friends.

2. Keywords (Glossary)

Event: An event represents a quiz event that is taking place in a specific location (pub, café etc.). It includes information about the quiz event including the start time and the location.

Exp: Short for experience, Exp points are given to users when they participate in quiz events and answer a question correctly. In other words, the Exp of a user shows how many quiz events that user participated in. We advise considering users with higher Exp for your team.

Group: A group (that is formed by a user) represents a group of maximum five users that belongs to a specific event.

Host: Host is the location where an event will take place. A host can be a pub, a café or anywhere public with tables and chairs where people can sit together and solve quiz questions.

Rank: All users of GrouPub are ranked based on their Exp points. The rank of a user indicates the position of that user in the leaderboard.

Title: Depending on their Exp points, each user is given a title, starting from 'Novice' all the way up the 'Legend'.

User: A user of GrouPub with a valid account.

3. Introduction

People spend the majority of their time in school or in workplace. Since this is the case there is a little time left to socialize with others, finding new people to chat or spending some quality time with the friends or colleagues. Our application, GrouPub, will help people in these manners. GrouPub enables its user to find new friends, spend a good time with friends in various places and makes people more social through organizing various quiz nights with diverging quiz themes in different places such as cafes and bars.

GrouPub is a unique application since there is no application that covers the concept of chatting with others in a specified area and creating fun quiz environments in different locations with multiple users to compete with. The user sees the events, quiz nights, in his/her smartphone and can join to those events with a single button click. The user also can create groups or can join existing groups and can meet with different people easily. Since the quiz nights held in different places, the user also able to discover new places.

The user can communicate with any people that enable group discussion for individuals in a common location. Mobile user can turn-on and run the client application, basically. The client application gets the physical location by QR code provided by place such as pubs, restaurants etc. When the mobile user wants to communicate with other users, client application gets the location information and relevant parameters from mobile device to server to register the communication that is within a valid common area. After the communication is successfully registered, a confirmation request is sent from server back to the mobile device [1].

It is easy to apply this application to any cafes or bars since it only requires the owner to prepare the QR codes needed to verify the people who participates in the quiz night in that café or bar. There is a chance to involve commerce in GrouPub since with this application the café and bars get more customers in the quiz nights which means there can be an agreement with these places and can schedule quiz nights in these places accordingly.

GrouPub is a social application provides user private or group conversation within a valid common area by mobile phone users in a virtual chat room and quiz game interface. The rest of the report is organized as follows: Section 1 clarifies the application with its constraints and professional and ethical issues within an application. Section 2 describes the functional and non-functional requirements in software development and finally Section 3 concludes the specifications of application.

4. Purpose of the Project

The purpose of GrouPub is to create a social environment where even strangers can form groups, share a few drinks and socialize while trying to win a quiz event. It is an alternative activity that saves users from their daily routine, refresh their minds, make some new friends and just have fun. Everyone needs something to relax and socialize with other people and GrouPub provides its users an easy way to relax and a fun night with friends or strangers.

5. Final Architecture and Design

5.1. Subsystem Decomposition

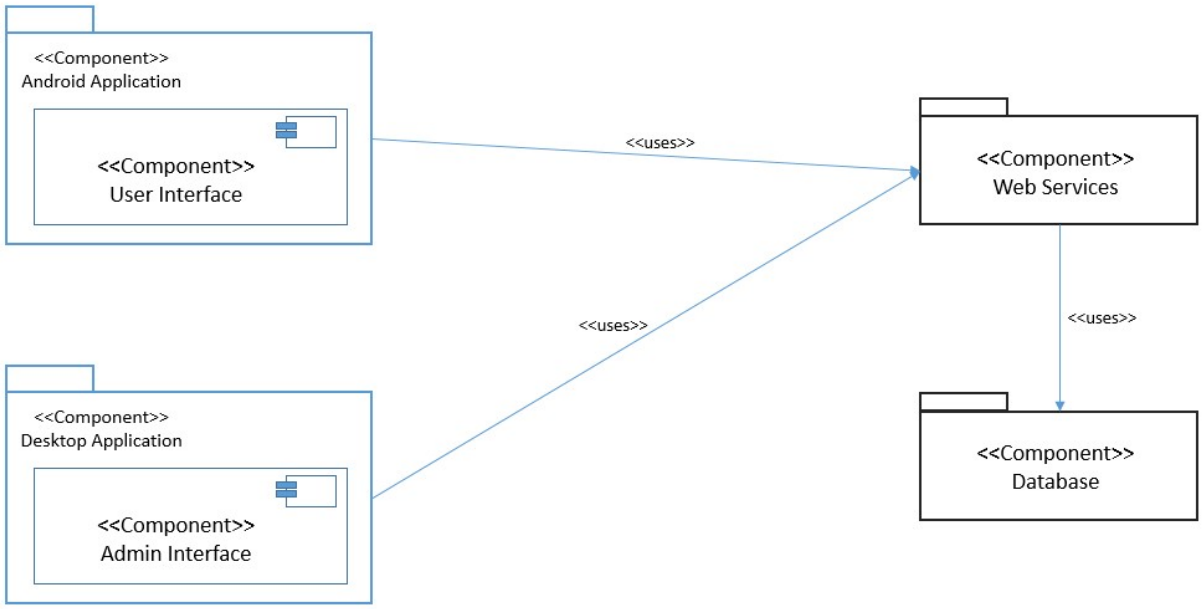


Figure 1 - Subsystem Decomposition

The android application has two subcomponents; the user and the admin interface. These components are implemented using Android Studio. By using a web container, these two components will connect to the server which is implemented in Spring MVC.

The database manager is the controller of information that is stored in our database. Some of the data's stored in our database are as follows (see the section Persistent Data Management for the full list); the ID, name and location of an event, the ID's and passwords of users, the ID's and participants of groups and so on.

The web service provides the connection between the application server and the database. In addition, it provides all the core functionalities of our application like registration of users, events or locations, the user interface of the quiz, the leaderboards, the map (to easily locate the event) and basically each page you see in the application.

5.2. Hardware/Software Mapping

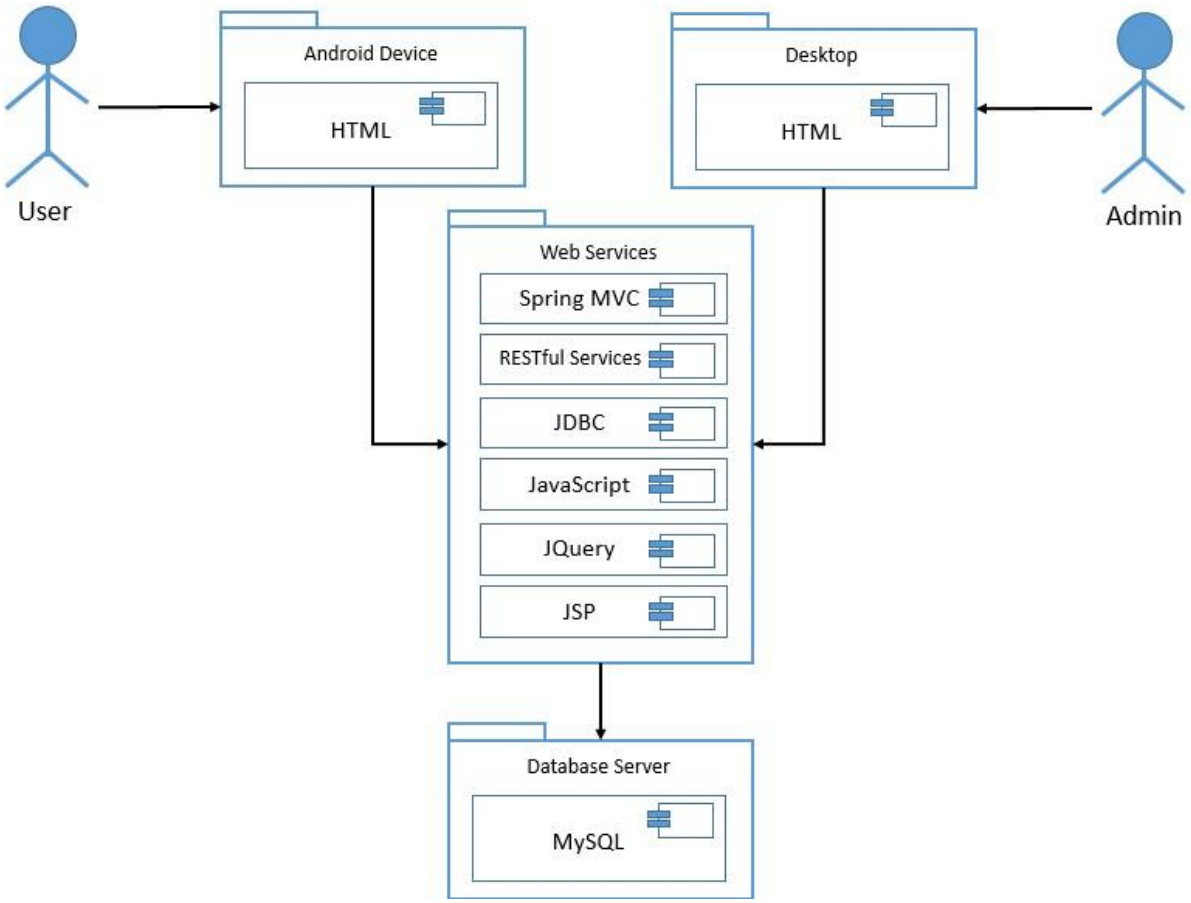


Figure 2 - Hardware/Software Mapping

Users will connect to the application by using an android device while the admins can connect to the application using their PC. Both the users and the admins will use the web services to connect to the database and insert/delete information. For the web services we use Spring MVC, RESTful Services, JDBC, JavaScript, JQuery and JSP. The database server is using MySQL.

5.3. Persistent Data Management

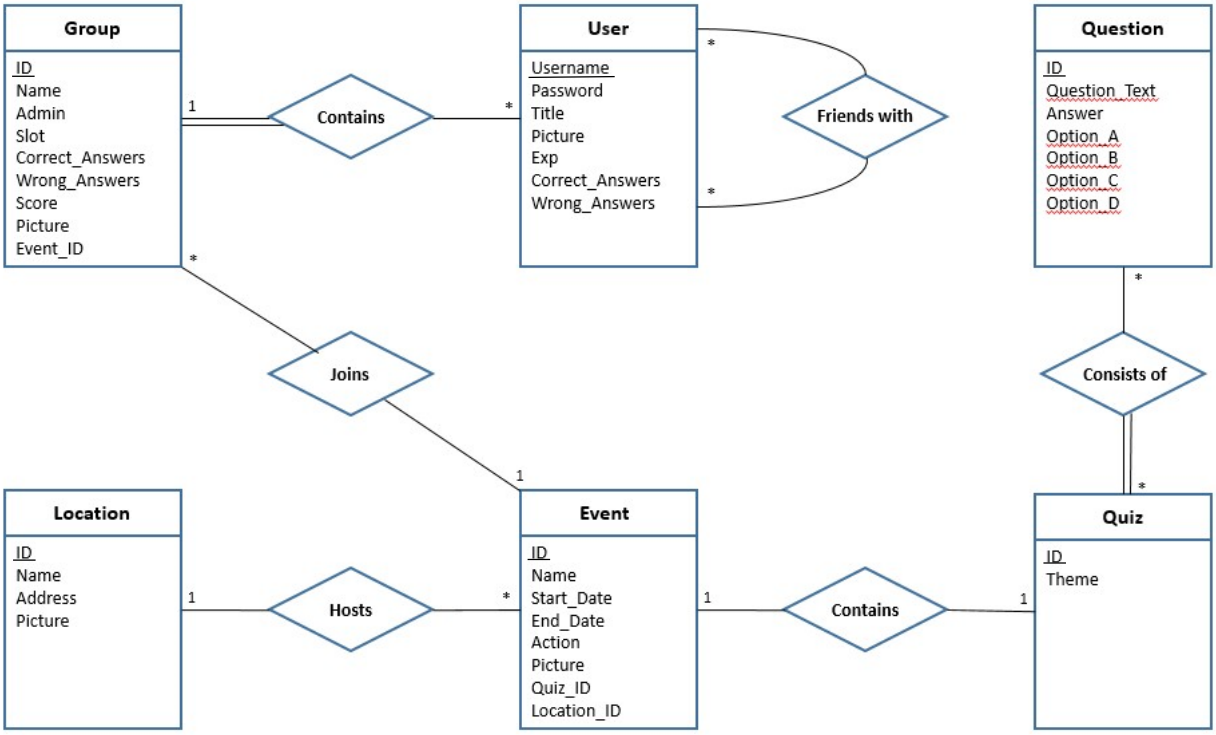


Figure 3 - ER Diagram

The ER diagram above shows the main structure of our application. Users can form groups and can become friends with other users. Groups can join events. A location hosts an event. An event contains a quiz and a quiz consists of questions. All attributes of these entities are stored in our database exactly as shown in the ER diagram.

There are additional tables in our database which are not included in the ER diagram for simplicity. Those tables are shown separately below;

Figure 4 - Event_Code

This table holds the event codes necessary to participate in an event. Each code can only be used once by a user. When an event is created, a certain number of codes are generated randomly. After a user uses a code, that code will be deleted from this table. The event ID specifies which event the code belongs to.

Figure 5 - Event_Online_User

This table shows the online users in an event. After a user successfully enters an event using his/her event code, that user will be added to this table (by username) with the corresponding event ID.

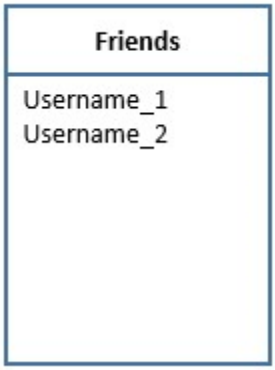


Figure 6 - Friends

This table shows the friendship relations between users. After User A and User B becomes friends, their usernames (Username_1 and Username_2) will be recorded here.

Figure 7 - Friend_Requests

In order to become friends with a user, a friend request must be sent first. When User A (Username_from) sends a friend request to User B (Username_to), this request will be recorded in this table. The reason for having this table is to be able to show the friend request notifications to users.

Figure 8 - Group_Answers

When a user (Username) that belongs to a certain group (Group_ID) answers a question, it will be stored in this table. Answer_Bool indicates whether the answer is true or false. This table used for group point calculations.

Figure 9 - Group_Flag

This table shows whether all members of a group (Group_ID) has answered the current question or not. If the Flag is false, then some users have not answered the current question yet. The reason for having this table to prevent incorrect group point calculation during quizzes.

Figure 10 - Group_Online_Count

This table shows the number of online users (Online_Count) that are present in an event in a specific group (Group_ID).

Figure 11 - Group_Requests

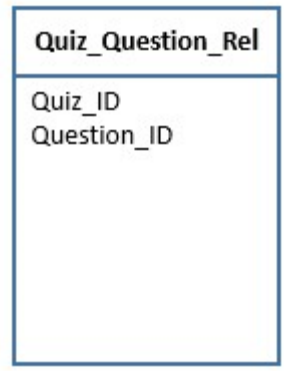
If a user (Username) receives an invitation to a group (Group_ID), it will be shown here. The reason for having this table is to be able to show group requests as notifications to our users.

Figure 12 - Group_User_Rel

This table (group user relation) shows the members (Username) of groups (Group_ID).

Figure 13 - Messages

This table holds the messages (Message_Text) that is sent by users (Username) who belong to a certain group (Group_Name) during and event (Event_ID).



This table shows the questions (Question_ID) that belong to a quiz (Quiz_ID).

Figure 14 - Quiz_Quesion_Rel

6. Final Packages

6.1. Java Packages

6.1.1. Controller

This package contains the controller classes of our MVC project structure. Controllers are the RESTful services that handles the post and get requests from the JSP pages which are the view components of the project.

Controller
AdminController.java
EventController.java
GroupController.java
ImageController.java
LoginController.java
MainController.java
ProfileController.java
QuestionController.java

6.1.2. Model

6.1.2.1. DAO

This package contains the DAO's (Data Access Object). DAO's are the java classes that are responsible of fetching the desired data from the database by executing SQL queries using JDBC (Java Database Connectivity).

dao
EventDAO.java
GroupDAO.java
ImageDAO.java
LocationDAO.java
QuestionDAO.java
UserDAO.java

6.1.2.2. Entity

This package contains the entities that are called POJO's (Plain Old Java Object). POJO's contain only the attributes of the entities created in the database along with the getter and setter methods. DAO's are responsible of creating and filling the instances of these POJO's.

Entity
Event.java
EventEndTimer.java
EventStartTimer.java
Group.java
Location.java
Message.java
Question.java
Quiz.java
User.java

6.1.3. Exceptions

This package is created in order to contain developer created exceptions.

exceptions
UserAlreadyExistException.java
GroupNameAlreadyExistsException.java
UserAlreadyExistsException.java

6.2. Resource Package

Resource package contains the XML files that hold the necessary configurations for the project. At this very moment, our project needs only the following XML file that contains database connection credentials.

resources
datasource.xml
log4j.xml

6.3. Web Packages

Web package contains the necessary directories that contain files for view component of the MVC structure.

6.3.1. Resources

This package will contain the necessary resources for the web pages such as CSS, JS and image files.

6.3.1.1. CSS

css
changePassword.css
event.css
leaderboard.css
map.css
main.css
question.css
register.css
settings.css
users.css

6.3.1.2. Images

This directory will contain the image files and icons of our project. Initially, this directory will contain group_default.jpg and user_default.jpg, which are the default images for users and groups. When an image is set by a user, that image will be saved here under the format group_groupName.jpg if it is a group image or user_Username.jpg if it's a user image.

6.3.1.3. JS

This directory will contain the necessary JavaScript functions in order to be used in the web pages. Not all JS files can be determined before creating all of the web pages, however the following are the currently available ones.

js
countdown.js
jquery.js
menu.js
questionResult.js

6.3.2. WEB-INF

WEB-INF is a directory that contains files that are hidden to any user. The files are only available to the controllers for the server-side tasks.

6.3.2.1. Pages

This directory contains the JSP files which are the views of the MVC structure. Those JSP files are shown to users via the controllers' supervision. (because of the considerable amount of .jsp files, they are shown in a grid layout rather than a single column)

pages		
aboutUs.jsp	eventInAction.jsp	question.jsp
admin.jsp	evetnResult.jsp	questionResult.jsp
aGroup.jsp	groups.jsp	register.jsp
allEvents.jsp	leaderboard.jsp	registerValidation.jsp
allGroupsInEvent.jsp	login.jsp	requests.jsp
changePassword.jsp	logout.jsp	requestsInAction.jsp
changeUsername.jsp	main.jsp	settings.jsp
createGroup.jsp	map.jsp	userProfile.jsp
error.jsp	myFriends.jsp	userProfileBackTo.jsp
event.jsp	myGroup.jsp	userProfileInAction.jsp
eventCode.jsp	myProfile.jsp	userSearch.jsp

7. Final Class Diagram

The following chart shows the relationships among classes used in GrouPub. Please note that the classes shown in this chart are not in full detail, only their class names are shown for the sake of simplicity. The detailed class interfaces are given after the diagram.

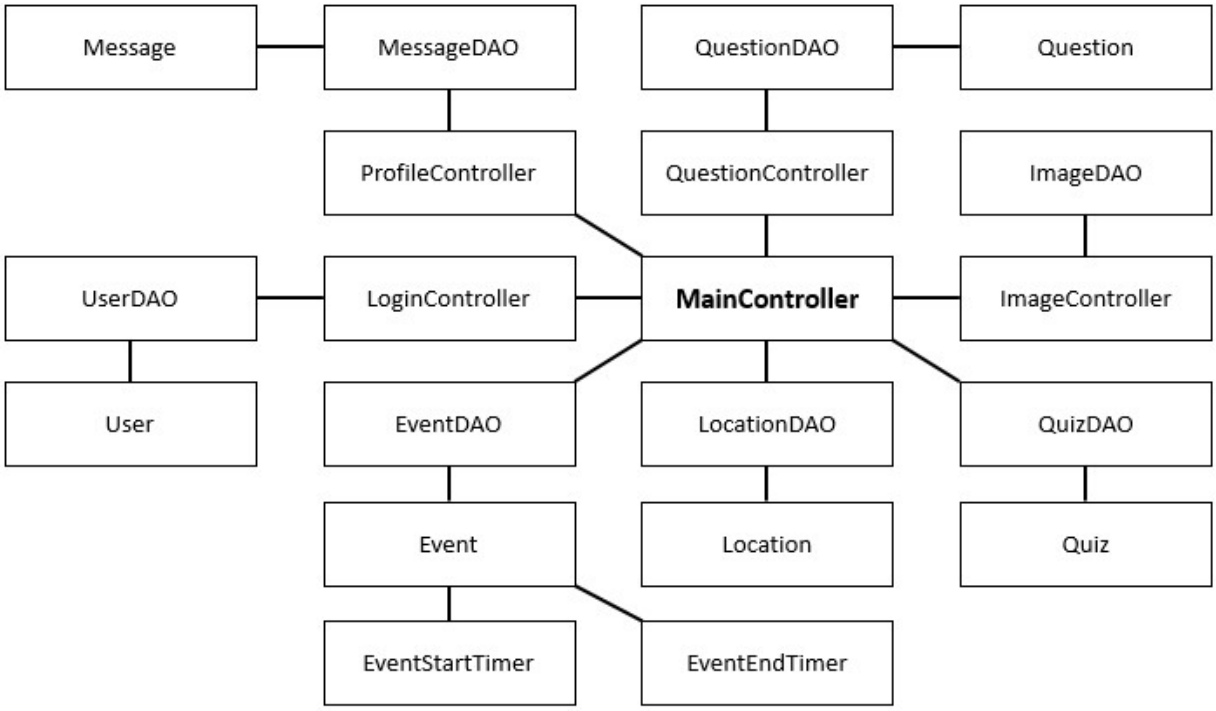


Figure 1155: GrouPub MVC Structure Class Diagram

Event
This class gets and sets information about quiz event such as address of it and when it will start.
-int id -Date startDate -Date endDate -String name -boolean action -boolean picture -int locationId -int quizId
+getId(): int +setId(int id): void +getStartDate(): Date +setStartDate(Date date): void +getEndDate(): Date +setEndDate(Date date): void +getName(): String +setName(String name): void +getLocationId(): int +setLocationId(int locationId): void +getQuizId(): int +setQuizId(int quizId): void +isPicture(): bool +setPicture(boolean action): void +isAction(): bool +setAction(boolean action): void

EventEndTimer
This class is used to end the timer in quiz question page.
+Timer timer +int eventId
+EventEntTimer(int eventId, Date date): void

EventStartTimer
This class is used to start the timer in quiz question page.
+Timer timer +int eventId
+EventStartTimer(int eventId, Date date): void

Group
This class represents the groups formed by the users for quiz events.
-int id -String name -String admin -int slot -int wrongAnswers -int correctAnswers -boolean picture -int eventId -double score
+getId(): int +setId(int id): void +getName(): String +setName(String name): void +getAdmin(): String +setAdmin(String admin): void +getSlot(): int +setSlot(int slot): void +getCorrectAnswers(): int +setCorrectAnswers(int correctAnswers): void +getWrongAnswers(): int +setWrongAnswers(int wrongAnswers): void +getEventId(): int +setEventId(int eventId): void +isPicture(): bool +setPicture(boolean action): void +getScore(): double +setScore(double score): void

Location
To keep track of the location that are registered to GrouPub, this class is used.
-int id -String name -String address -boolean picture
+getId(): int +setId(int id): void +getName(): String +setName(String name): void +getAddress(): String +setAddress(String address): void +isPicture(): bool +setPicture(boolean action): void

Message
This class represent the messages texted by users during events.
-int id -String messageText -int eventId -String groupName -String userName
+getId(): double +setId(double id): void +getMessageText(): String +setMessageText(String messageText): void +getEventId(): int +setEventId(int eventId): void +getGroupName(): String +setGroupName(String groupName): void +getUsername(): String +setUsername(String username): void

Question
To get the question, choices and the correct answer of the question, this class is used.
-int id -String questionText -String answer -String optionA -String optionB -String optionC -String optionD
+getId(): int +setId(int id): void +getQuestionText(): String +setQuestionText(String questionText): void +getOptionA(): String +setOptionA(String optionA): void +getOptionB(): String +setOptionB(String optionB): void +getOptionC(): String +setOptionC(String optionC): void +getOptionD(): String +setOptionD(String optionD): void

Quiz
An event stores a quiz which stores questions. Every quiz has its own id to identify each of them in terms of scores.
-int id
+getId(): int +setId(int id): void

User
This class is the generic model that are used in both server and client.
-String username 'String password -String title -boolean Picture -double exp -int correctAnswers -int wrongAnswers
+getPassword(): String +setPassword(String password): void +getUsername(): String +setUsername(String username): void +getTitle(): String +setTitle(String title): void +getExp(): double +setExp(double exp): void +getCorrectAnswers(): int +setCorrectAnswers(int correctAnswers): void +getWrongAnswers(): int +setWrongAnswers(int wrongAnswers): void +isPicture(): bool +setPicture(boolean action): void

AdminController
The controller class for the admin page.
+getAdminPage(int pass): String +setEventPage(int eventId, String eventName, String startDate, String endDate, int quizId int locationId, Model model): String +deleteEvent(int eventId, Model model): String

EventController
The controller class for the events in GrouPub.
+getEventPage(int id, boolean main, Model model, HttpSession session): String +printEventCodePage(int eventId, boolean main, Group myGroup, String username, HttpSession session): String +checkEventCode(int eventId, boolean main, Group myGroup, int code, String username, Model model, HttpSession session): String +prepareEventInActionPage(int eventId, boolean main, Group myGroup Model model, HttpsSession session): String +getEventInActionPage(int eventId, boolean main, Event event, ArrayList<Group> groupArr, ArrayList<Question> questionArr, Group myGroup Model model, HttpsSession session): String +printQuestion(ArrayList<Question> questionArr, int questionNum, int questionTimeLeft, Model model, HttpsSession session): String +printQuestion(String answer, myAnswer, double myTime, Group myGroup, Model model, HttpsSession session): String +allGroupsInEventPage(ArrayList<Group> groupArr, int eventId, Model model, HttpsSession session): String +getChatInJSON(int eventId): ArrayList<Message> +sendChatInJSON(String msg, int eventId, Group myGroup, String username): void -calculateGroupAnswer(int groupId): void synchronized +forceClose(Group myGroup): void

GroupController
The controller class for the groups in GrouPub.
+getMyGroupPage(int groupId, int eventId, boolean main, Model model, HttpsSession session): String +getCreateGroupPage(String groupName, int slot, int eventId, Model model, HttpsSession session): String +deleteGroup(int groupId, eventId, Model model, HttpsSession session): String +getAGroupPage(int groupId, int eventId, boolean main, Model model, HttpsSession session): String +requestAGroupPage(int groupId, int eventId, boolean main, boolean request, HttpSession session): String +seeGroupRequests(int groupId, int eventId, boolean main, HttpSession session): String +seeGroupRequestsInAction(int groupId, int eventId, boolean main, HttpSession session): String +evaluateGroupRequests(int groupId, int eventId, boolean accept, String username, Boolean main, Model model, HttpSession session): String +evaluateGroupRequestsInAction(int groupId, boolean accept, String username, int eventId, boolean main, Model model, HttpSession session): String +kickUser(int groupId, int eventId, String username, boolean main, Model model, HttpSession session): String +getGroupsPage(int eventId, boolean main, Model model, HttpSession session): String +getMapPage(int groupId, int eventId, boolean main, Model model, HttpSession session):

ImageController
The controller class for the group and user images in GrouPub.
+setProfilePicture(MultipartFile upfile, Model model, HttpSession session): String +setGroupPicture(int groupId, int eventId, boolean main, MultipartFile upfile, Model model, HttpSession session): String +removeProfilePicture(Model model, HttpSession session): String +removeGroupPicture(int groupId, int eventId, boolean main, Model model, HttpSession Session): String

LoginController
This class is responsible for authentication to the application via password.
+getLoginPage(ModelMap model): String +checkCredentials(String username, String password, ModelMap model): String +getLogoutPage(HttpSession session): String +getRegisterPage(): String +addUser(String username, String password, String passwordConfirm, Model model): String +getAboutUsPage(): String

Main Controller
Main Controller is the class that controls every user action made after logging in. Therefore, it is the core class that binds all other controllers to each other.
+getMainPage(Model model, HttpSession session): String +getSettingsPage(HttpSession session): String +getChangePasswordPage(HttpSession session): String +setPassword(String oldPassword, String newPassword, String confirmPassword, Model model, HttpSession session): String +getLeaderboardPage(Model model, HttpSession session): String +getChangeUsernamePage(Model model, HttpSession session): String +setUsername(String newUsername, Model model, HttpSession session): String +getAllEventsPage(Model model, HttpSession session): String

Profile Controller
The controller class for the profiles of our users.
+getMyProfile(Model model, HttpSession session): String +getUserProfileBackTo(String username, int groupId, int eventId, boolean main, int backTo, Model model, HttpSession session): String +addFriendBackTo(String friendUsername, int todo, int backTo, int groupId, int eventId Boolean main, Model model, HttpSession session): String +getMyFriends(Model model, HttpSession session): String +getMyFriendsForGroup(int groupId, int eventId, boolean main, Model model, HttpSession session): String +getMyFriendsToGroup(int groupId, String friendUsername, int eventId, boolean main, Model model, HttpSession session): String +getUserProfile(String username, boolean backToFriends, Model model, HttpSession Session): String +getUserSearch(Model model, HttpSession session): String +addFriend(String friendUsername, int todo, Model model, HttpSession session): String +getUserProfileInAction(String Username, int groupId, int eventId, boolean main, Model model, HttpSession session): String +addFriendInAction(String friendUsername, int todo, int groupId, int eventId, boolean Main, Model model, HttpSession session): String

EventDAO
Data access object that provides an abstract interface to event database.
-DataSource dataSource
+setDataSource(DataSource dataSource): void +getAllEvents(): ArrayList<Event> +getLast3Events():ArrayList<Event> +getEventById(int id): Event +setEventPicture(int eventId): void +deleteEventPicture(int eventId): void +startEvet(int eventId): void +endEvent(int eventId): void +addEvent(int eventId, String eventName, String startDate, String endDate, int quizId, Int locationId): void +getEventIdByCode(int code): int +removeCode(int eventId, int code): void +addEventOnlineUser(int eventId, String username): void +isUserInEvent(int eventId, String username): boolean +addCodes(int eventId, ArrayList<Integer> codeArr): void +removeEventById(int eventId): void

GroupDAO
Data access object that provides an abstract interface to group database.
-DataSource dataSource
+getGroupsByEventId(int eventId): ArrayList<Group> +getGroupByUsernameInEvent(String username, int eventId): Group +getGroupById(int id): Group +addGroup(String groupName, String admin, int slot, int eventId): void +getGroupByName(String name): Group +addUserToGroup(int groupId, String username): void +getUsersByGroupId(int groupId): ArrayList<User> +deleteGroupById(int groupId): void +deleteFromGroupRequests(int groupId, String username): void +getGroupRequests(int groupId): ArrayList<User> +setGroupRequest(int groupId, String username): void +isGroupRequestSent(int groupId, String username): boolean +kickUserFromGroup(int groupId, String username): void +setGroupPicture(int groupId): void +deleteGroupPicture(int groupId): void +addCorrectAnswerToGroupPool(int groupId, String username): void +addWrongAnswerToGroupPool(int groupId, String username): void +getAnswerSumFromGroupPool(int groupId): int +getAnswerCountFromGroupPool(int groupId): int +deleteGroupFromGroupPool(int groupId): void +addCorrectAnswerToGroup(int groupId): void +addWrongAnswerToGroup(int groupId): void +incrementOnlineCount(int groupId): void +decrementOnlineCount(int groupId): void +getOnlineCount(int groupId): int +setGroupFlag(int groupId, boolean flag): void +getGroupFlag(int groupId): boolean +addScore(int groupId, double score): void +getScore(int groupId): double

ImageDAO
Data access object that provides an abstract interface to image database.
-DataSource dataSource
+setDataSource(DataSource dataSource): void +addUserImage(byte[] imageInByte, String webappRoot, String username): void +removeUserImage(String webappRoot, String username): void +addGroupImage(byte[] imageInByte, String webappRoot, String groupName): void +removeGroupImage(String webappRoot, String groupName): void

LocationDAO
Data access object that provides an abstract interface to location database.
-DataSource dataSource
+setDataSource(DataSource dataSource): void +getLocationById(int id): Location +setLocationPicture(int locationId): void +deleteLocationPicture(int locationId): void

MessageDAO
Data access object that provides an abstract interface to message database.
-DataSource dataSource
+setDataSource(DataSource dataSource): void +getMessagesByEventId(int eventId): ArrayList<Message> + addMessage(double msgId, String groupName, String userName, String msgText, int eventId): void

QuestionDAO
Data access object that provides an abstract interface to question database.
-DataSource dataSource
+setDataSource(DataSource dataSource): void +findByQuestionId(int questionId): Question +findAllQuestionsByQuizId(int quizId): ArrayList<Question>

UserDAO
Data access object that provides an abstract interface to user database.
-DataSource dataSource
+setDataSource(DataSource dataSource): void +addUser(String username, String password): void +getUser(String username): User +setPassword(String username, String password): void +setUsername(String oldUsername, String newUsername): void +getAllUsers():ArrayList<User> +isFriend(String username1, String username2): boolean +addFriend(String username1, String username2): void +deleteFriend(String username1, String username2): void +addFriendRequests(String username1, String username2): void +deleteFriendRequests(String usernameFrom, String usernameTo): void +isFriendRequestSent(String usernameFrom, String usernameTo): boolean +isFriendRequested(String usernameFrom, String usernameTo): boolean +getAllFriends(String username): ArrayList<User> +setUserPicture(String username): void +deleteUserPicture(String username): void +setExp(String username, double exp): void +incrementCorrectAnswers(String username): void +incrementWrongAnswers(String username): void

8. Final Status

Web application development is successfully completed except minor bugs. In addition to this, because mobile application development is triggered by web-container, which is the component of a web-server that interacts with Java servlets and responsible for mapping a URL to a particular servlet and ensuring that the URL requester has the correct access [2], our mobile application development is also completed.

9. Engineering Solutions and Contemporary Issues

GrouPub is a location-based quiz application that provides user-friendly interface to reach all kinds of users. It enables users to improve their general knowledge via provided quiz application with variety of question themes from geography to science. User-friendly interface aims to reach various users regardless of their gender, age, culture and so on.

It is development language is Turkish because of the target users of the application. We aim to put it in a market with its first version with Turkish interface because its users will be in the Turkey for now.

The authentication way of GrouPub application is approving users via their usernames and passwords. User information is stored in private database so that it's secure and protected in terms of user privacy.

To enter to the quiz application, user need to some information/code for authentication. We will generate numerous unique "quiz codes" when event about to start. Each user will go into the application via these codes, then the validity of that quiz code, which is assigned to that user, will be removed from the quiz code table from the database. The uniqueness of the quiz code will be protected so that no other user use that code again.

In addition, only necessity for users to have is Internet connection. Because smartphones have mobile data and the places may provide Wi-Fi connection, it is easy to reach for the users.

10. Tools and Technologies Used

In this section, report provides detailed information about which tools and technologies we used through the development process of GrouPub application.

Programming Languages

We used **JSP** (JavaServer Pages) to implement dynamic web pages. It uses Java programming language and is based on HTML document types. JSP is used independently for the server side of the model-view-controller design of the GrouPub application. In addition, we use Spring Framework for the Java application.

Web Technologies: **HTML**, **CSS** used for the web development so for the view side of the model-view-controller design.

Technologies, Libraries and Frameworks

Because we implement the application based on model-view-controller design, we used **Spring MVC** and **RESTful API** to handle HTTP requests and responses for the client-server communication.

We used **Ajax** and **JavaScript** to send a request to a server. Ajax enables updating the page without reloading it so it provides quick refresh for the web page.

IDE: **IntelliJ Idea**

Version Control: **GitHub**

It is web-based repository hosting service. We did source code management and repository checks via GitHub.

Databases/Data storage tools: **MySQL**

We used MySQL as a database for our web application GrouPub. It works as cross-platform and can be built and installed manually from source code.

Application and Web Servers: **Apache Tomcat 7**, **Jetty 8**

To deploy our final web application we used Tomcat 7 as the application server and Jetty 8 is used for testing purposes [2].

11. Resource Usage

This part includes main web resources we used through the implementation process. While developing web-application, we used tutorials provided in JSP tutorial resources [3]. We used HTML, CSS, JavaScript and Ajax tools to handle http requests in terms of model-view-controller design so to implement web-application interface with the help of tutorials provided in related resources [4]. We get the Spring MVC Framework usage information from the resources provided in such tutorial pages [5].

After completing web-application implementation, we use web application's URL to run the program as mobile application. In that point, we used web-container so the related tutorials are found in oracle java pages [6].

12. Similar Applications

There are bunch of applications that are similar to GrouPub. However, none of them is exactly the same with GrouPub. Let us mention several most similar ones.

- **QuizUp** [7]
    - **Similarities:**
        - Different themes for questions
        - World-wide user scores (leaderboard)
        - Player to player chat engine
        - Cannot be played offline (single player)
    - **Differences:**
        - QuizUp is not location-based
        - QuizUp does not provide team vs team challenges
        - No team chat rooms in QuizUp
        - No jokers in QuizUp

- **MoviePop** [8]
    - **Similarities:**
        - World-wide user scores (leaderboard)
        - Cannot be played offline (single player)
    - **Differences:**
        - MoviePop does not provide team vs team challenges
        - No chat engine in MoviePop
        - No rewards in MoviePop
        - MoviePop has a fixed question theme

- **Trivia Burst** [9]
    - **Similarities:**
        - World-wide user scores (leaderboard)
        - Different themes for questions
    - **Differences:**
        - Trivia Burst can be played offline (single player)
        - Trivia Burst does not provide team vs team challenges
        - No chat engine in Trivia Burst
        - No rewards in Trivia Burst

- **Quizoid** [10]
    - **Similarities:**
        - Different themes for questions
        - World-wide user scores (leaderboard)
    - **Differences:**
        - No chat engine in Quizoid
        - No rewards in Quizoid
        - Quizoid can be played offlline (single player)

- **Urban Quiz** [11]
    - **Similarities:**
        - Location-based
        - Different themes for questions
        - Cannot be played offline (single player)
    - **Differences:**
        - In Urban Quiz users creates the questions
        - No rewards in Urban Quiz

        - No chat engine in Urban Quiz
- **Pub Quiz** [12]

    - **Similarities:**
        - World-wide user scores (leaderboard)
        - Different themes for questions
    - **Differences:**
        - Pub Quiz is not location-based
        - Pub Quiz does not provide team vs team challenges
        - No chat engine in Pub Quiz
        - No rewards in Pub Quiz
        - Pub Quiz can be played offline (single player)

13. The Innovation We Provide

As seen in section 12, most of the quiz applications currently available are not location-based. Users can participate in quiz events from anywhere, most probably from their homes (for a typical user). This situation (just like any other application) prevents users from going out and socialize.

We would like to change this by encouraging our users to go out, socialize, group up and participate in quiz events together (which is why we named our application GrouPub instead of a name with the word 'quiz' in it). Unlike some popular application that creates addiction and traps their users at their homes, we would like to create a nice environment where users can play GrouPub and socialize at the same time.

14. Intuitive Flow of GrouPub

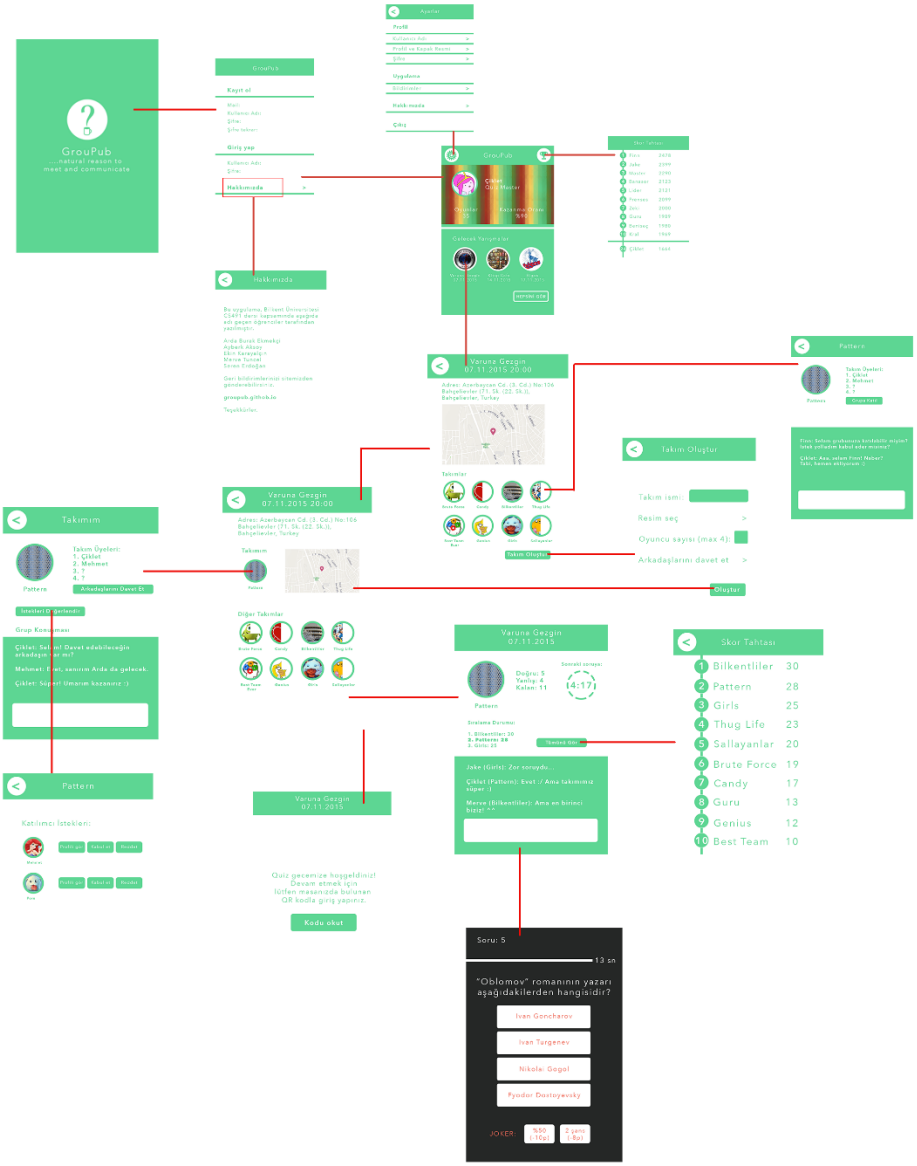


Figure 16 - Intuitive Flow of GrouPub

15. Conclusion

As the first thought of the development process, we were already close to idea of location-based application. However, we made minor changes of the project depending on the process. As the final form, we focused on the innovative aspects of the project such that while players are participating in a quiz event, they need to be in same location all at once. In addition to this, we aimed to develop a reliable application so GrouPub consistently performs according to its specifications. We wanted to variety of users so aimed to offer user-friendly interface so that every user regardless of their age would use it easily.

We are developing a technological and innovative software product that everyone wants to get involved. Because it provides entertainment, knowledge and also ability of communication among its users, our product may reach the large masses. We comprehend the importance of economic and technological concerns in software product market so that we are able to get feedback from GrouPub users and make some improvements in the current version of the application.

16. References

[1] Huang, Sheng Chao. Li, Ho Yin. “Location-Based Networking”. May 23, 2011

[2] https://en.wikipedia.org/wiki/Web_container

[3] http://www.tutorialspoint.com/jsp/

[4] http://www.w3schools.com/ajax/ajax_xmlhttprequest_send.asp

[5] http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

[6] http://docs.oracle.com/javaee/6/tutorial/doc/bnabo.html

[7] Play.google.com, 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.quizup.core&hl=en. [Accessed: 02- Jan- 2016].

[8] 2016. [Online]. Available: https://play.google.com/store/apps/details?id=air.com.freshplanet.games.MoviePop. [Accessed: 02- Jan- 2016].

[9] Play.google.com, 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.triviaburst.e5&hl=en. [Accessed: 02- Jan- 2016].

[10] Play.google.com, 2016. [Online]. Available: https://play.google.com/store/apps/details?id=de.habanero.quizoid&hl=en. [Accessed: 02- Jan- 2016].

[11] U. Quiz and T. Donder, "Urban Quiz on the App Store", *App Store*, 2016. [Online]. Available: https://itunes.apple.com/us/app/urban-quiz/id306263379?mt=8. [Accessed: 02- Jan- 2016].

[12] Play.google.com, 2016. [Online]. Available: https://play.google.com/store/apps/details?id=com.pubquiz. [Accessed: 02- Jan- 2016].

17. Appendix – User Manual

In order to get the GrouPub application, the user should enter the Android Application Store and download the application. When download is complete the android OS should install the application automatically. After the installation the user can enter the GrouPub main page (see Figure 17). To register, click 'Kayıt ol!'. You will be redirected to the registration page (see Figure 18). Just fill in the required form and hit 'Kayıt ol!'. Now you have successfully registered to GrouPub and can fully utilize the application.

Figure 17 - Main page of GrouPub

Figure 18 - Registration page of GrouPub