*Proiect: ,,Sport Event Application Fun&Health”*

Student: Casap Dumitru  
Grupa:30244

**1.Introduction**

Sport event application “Health&Fun” is a web application which will run on a server. The app will be available for all people who have an internet connection and it is proposed to people who like sport. This is a web application, so it can be accessed from any operating system. In the future, this app will be available for Android an iOS. In order to profit of benefits of this app the user should have an account. He should select the favorite sports and after that the user can see the sport events from the list of interests. He can see details about events, see the comments, going to an event, create a new event. There is possibility to chat with other users, to make new friends and to do sport together.

**2.Project requirements**

**2.1 Functional requirements**

The functional requirement specifies something the system should do. The application have three types of users: administrator and client which have an account and guest. All functional requirements which can be done by users on system are:

The regular user can perform the following operations:

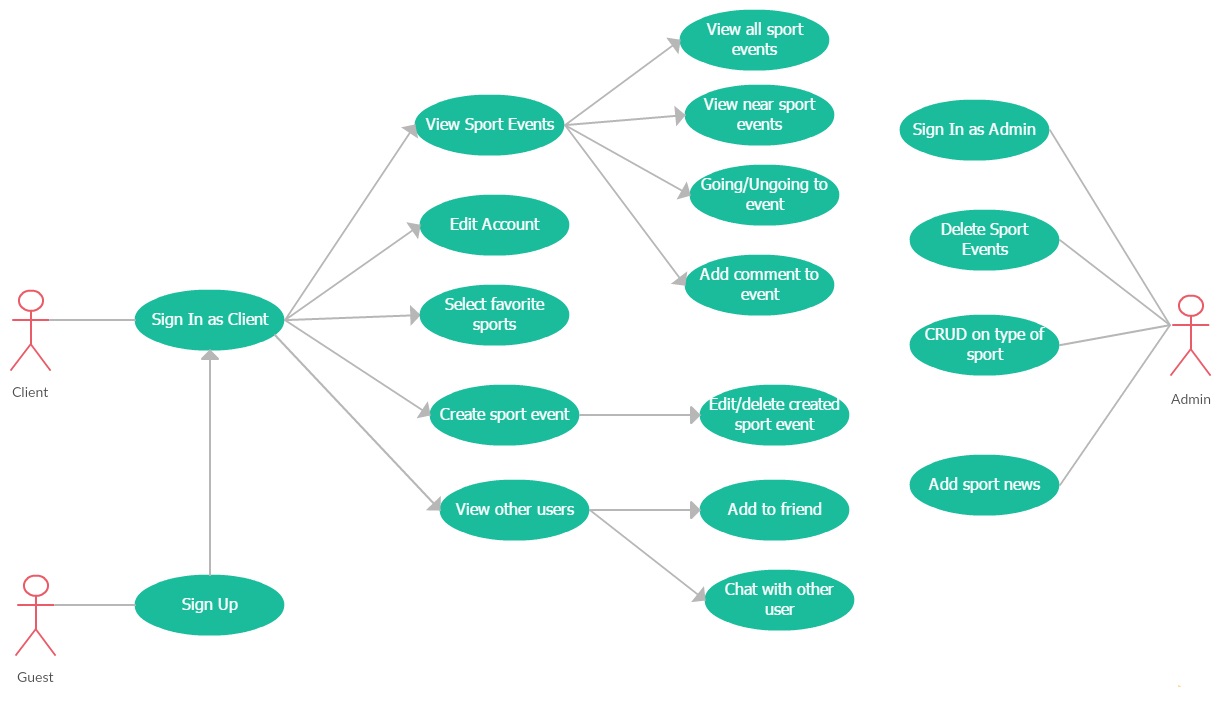
* Create a sport event
* View all/near sport events
* Update/delete his sport events.
* Edit his account.
* Select favorite sports.
* Create a sport event.
* Going/Ungoing to a sport event.
* Add comment to a sport event.
* Add to friend/ chat with other users

The administrator can perform the following operations:

* Delete sport events
* Delete accounts
* CRUD on type of sport
* Add sport news

**2.2 Non-functional requirements.**

* Availability  
   This app will be available for all persons who have internet access. This project has great intention for the future, so the server will be great, and it is estimated that this app can be accessed at the same time by 100000 of the persons. If the number of clients will grow, the server will be improved.
* Reliability  
   It is theoretically defined as the probability of succes ( 1 – Probability of Failre). The frequence of failures will depend of availability in this application. Realibility plays a key role in the cos-effectiveness of systems.
* Performance  
   Sport events have great intention on the market, and it is supposed that it will have more success. There are more stakeholder that will help to improve this idea. The app will be performance, with good speed of access and very confortable to use.
* Security  
   The application will be secure. The password will be encrypted and the account can be cracked . If the user will introduce the password wrong for five time, the account will be blocked and it can be restored by a link from email.
* Testability  
   The application have a team of testers, and every entity will be tested. The entire application will be tested in order to avoid the bugs on run time of application.
* Usability  
   The app is very easy and comfortable to use. The menu is very explicit and the user can simple to use the main functionalities of the app.

**3.Use-Case Diagram**

*Main use-cases describing the complex operation functionality:*

Use case: **Create Sport EventLevel: user-goal level**  
 Primary actor: Client  
 Main success scenario: Firstly, the client should login on application.  
 After that he has the possibility to create a sport event. The event should  
 have a name, a specific type of sport, a location, start and end date. After  
 creating it, the user have possibility to edit the sport event.

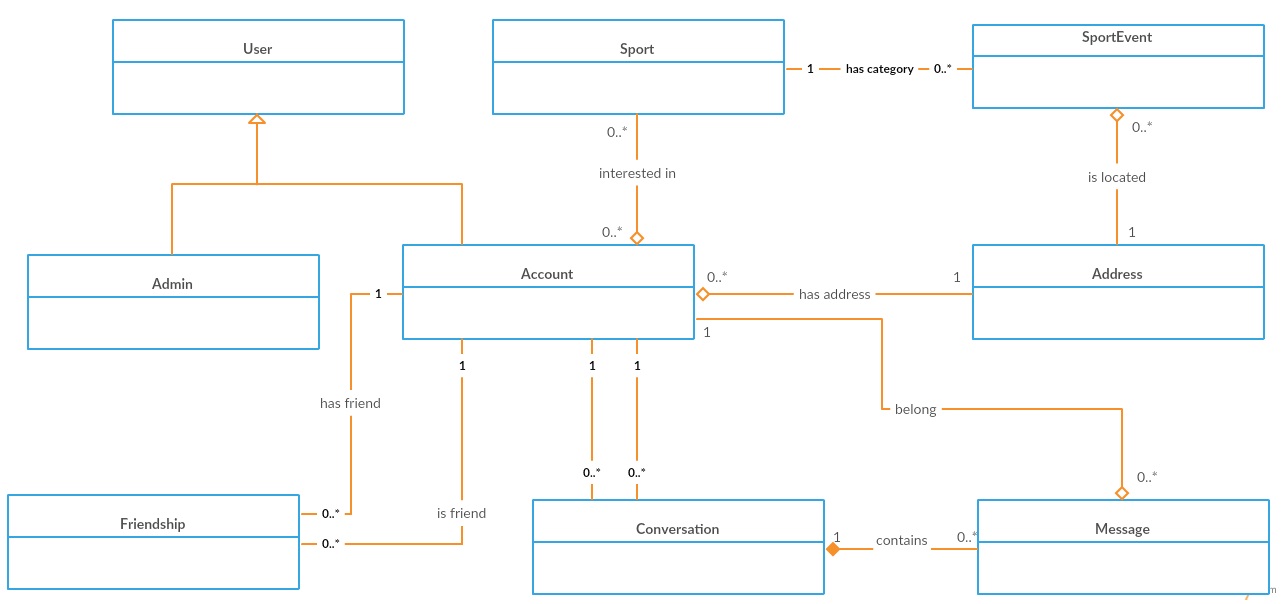
Use case: **Edit Account**  
 Level: user-goal level  
 Primary actor: Client  
 Main success scenario: Firstly, the client should login on application.  
 After that he has the possibility to edit his account. He can change name, password, email, location and in order to confirm the changes he must introduce his password.

Use case: **Add to friend**  
 Level: user-goal level  
 Primary actor: Client  
 Main success scenario: Firstly, the client should login on application.  
 After that he can search for a specific user by name. The client can view the important information about that user and can send a friend request.

**4. Elaboration**

**4.1 Domain Model**

A domain model is a system of abstractions that describes selected aspects of a sphere of knowledge. It is a representation of meaningful real-world concepts pertinent to the application that need to be modeled in software. In the case of this application, the domain model are represented by model classes : User, Admin, Account, Sport, SportEvent, Friendship, Conversation, Message, Address and relations between them. The main actor is account, which all be the client of this application. All models are related with Account model. The domain model of the sport event application “Health&Fun” are better explained by the conceptual class diagram.

****

**4.2 Conceptual Architecture**

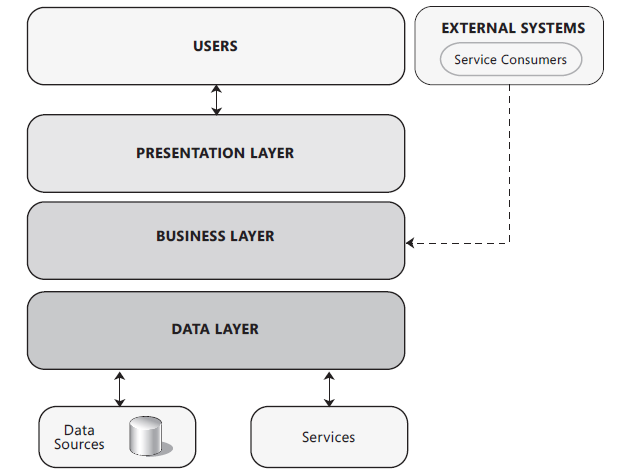
The architectural design pattern used in this project is the layers architectural pattern. An application can consist of a number of basic layers and the common three-layer design consists of the following layers:

**Presentation layer**–  is responsible for the delivery and formatting of information to the application layer for further processing or display. The presentation layer is the lowest layer at which application programmers consider [data structure](https://en.wikipedia.org/wiki/Data_structure) and presentation

**Business layer**– implements the core functionality of the system and encapsulates the relevant business logic

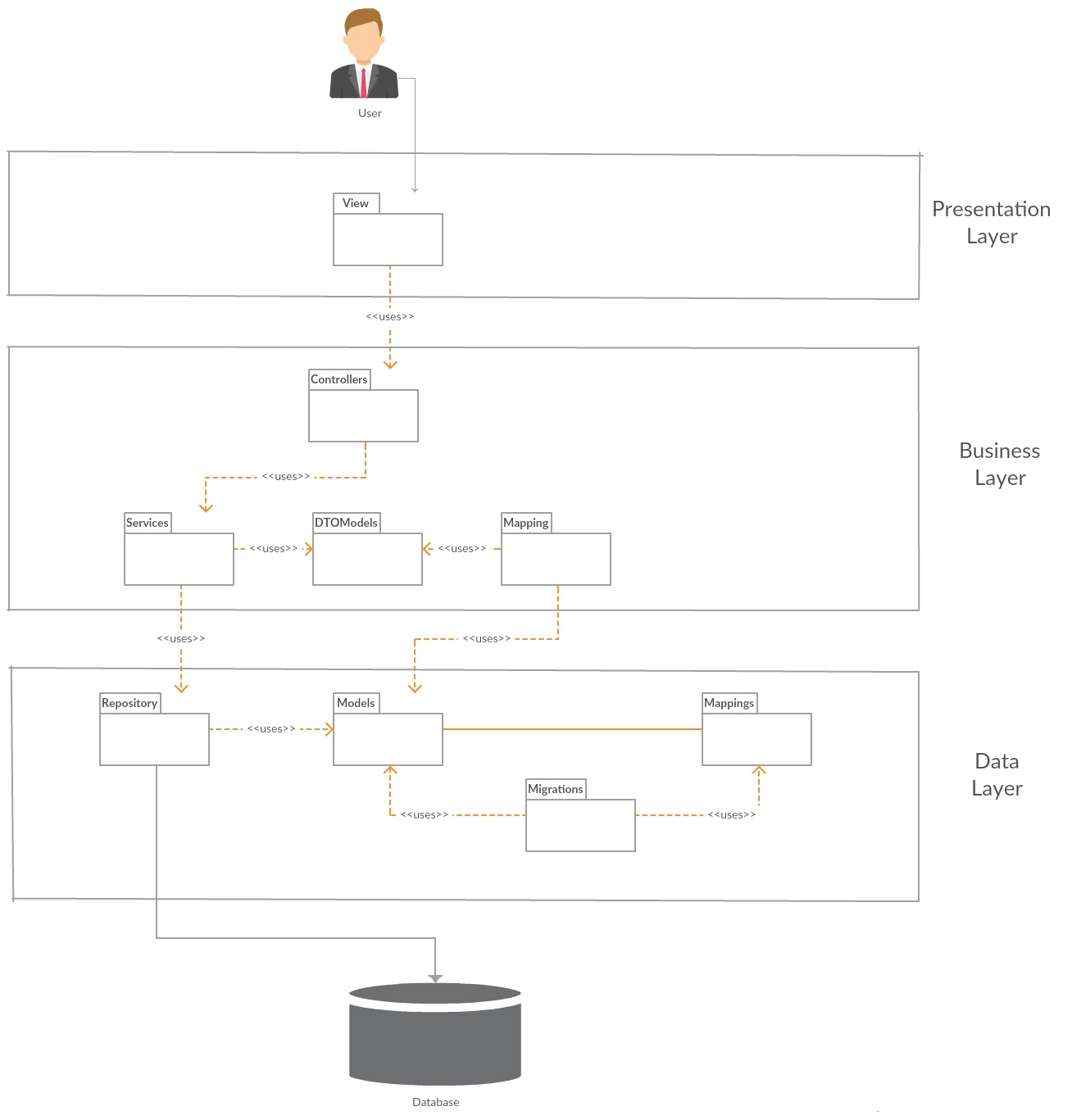
**Data layer** – provides access to data hosted within the boundaries of the system

Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application. Each layer in the architecture forms an abstraction around the work that needs to be done to satisfy a particular business request. For example, the presentation layer doesn’t need to know or worry about how to get customer data; it only needs to display that information on a screen in particular format. Similarly, the business layer doesn’t need to be concerned about how to format customer data for display on a screen or even where the customer data is coming from; it only needs to get the data from the persistence layer, perform business logic against the data ,and pass that information up to the presentation layer.

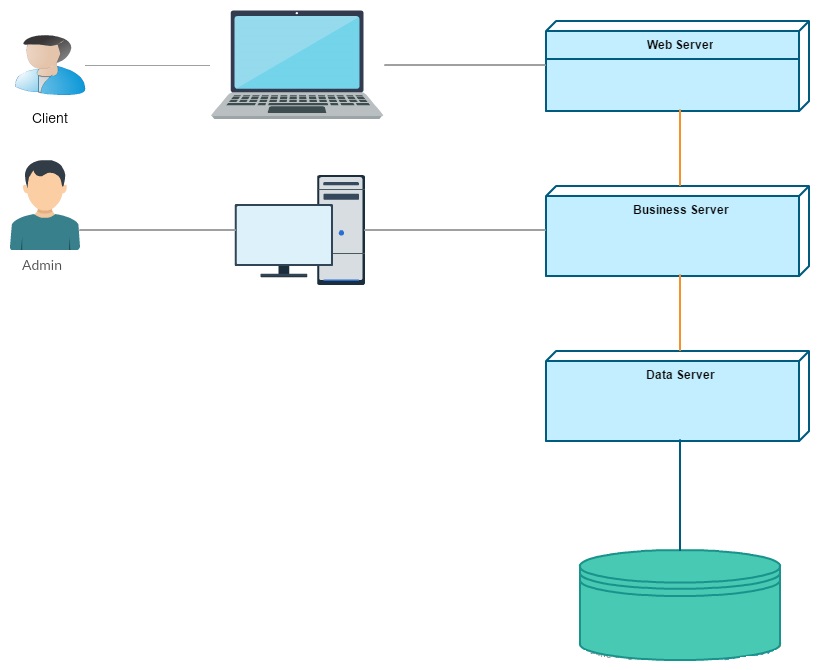


The layered architecture pattern is a solid general-purpose pattern, making it a good starting point for most applications, particularly when you are not sure what architecture pattern is best suited for your application.One of the powerful features of the layered architecture pattern is the separation of concerns among components.

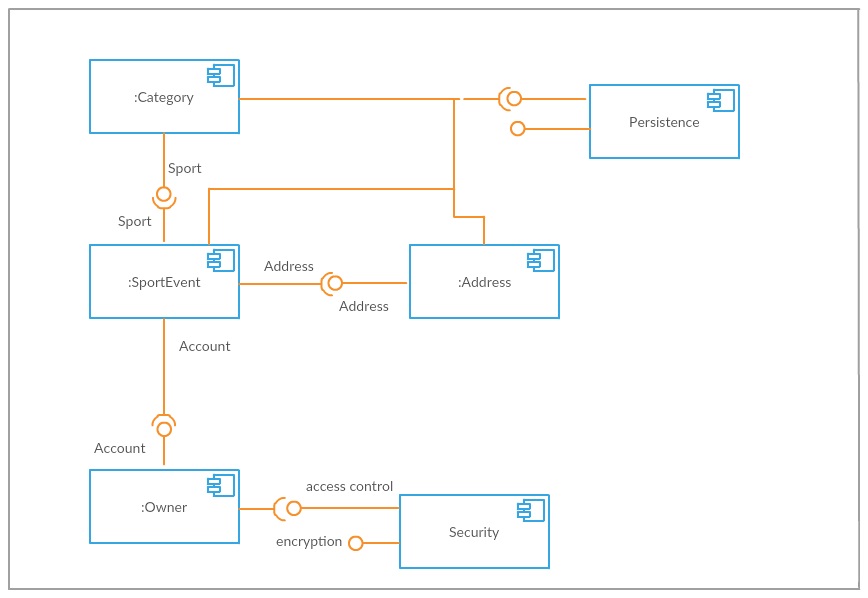
**4.3 Package Design**

****

**4.4 Deployment Diagram**

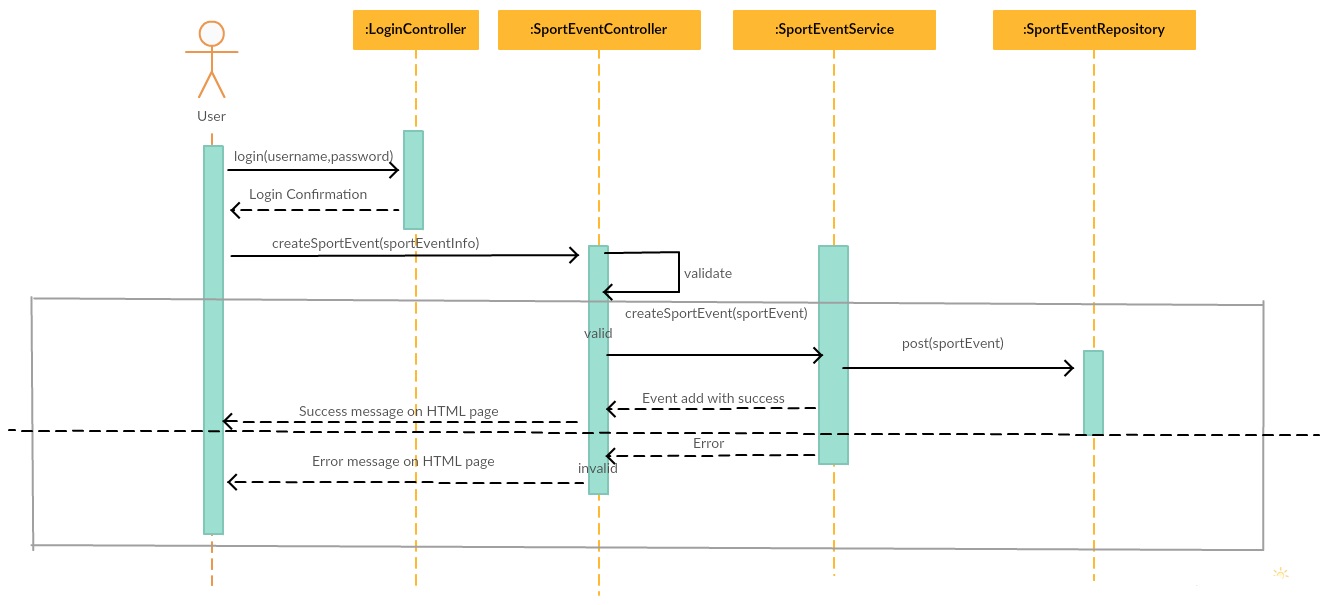
****

**4.5 Component Diagram**

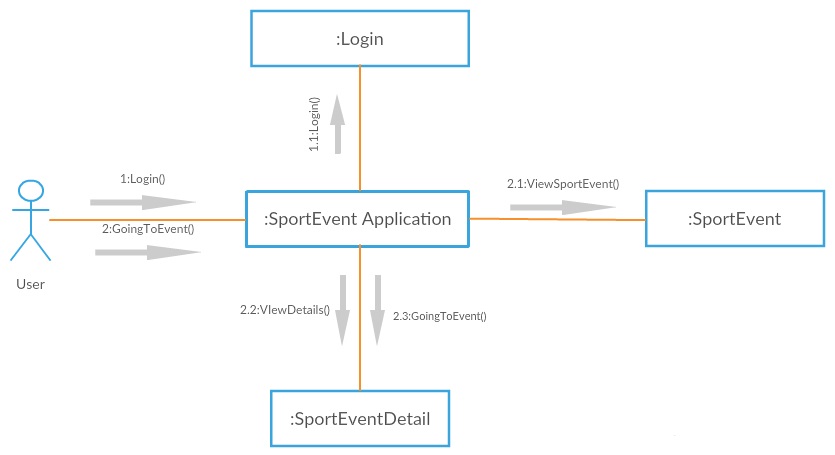
****

**4.6 Dynamic Behavior**

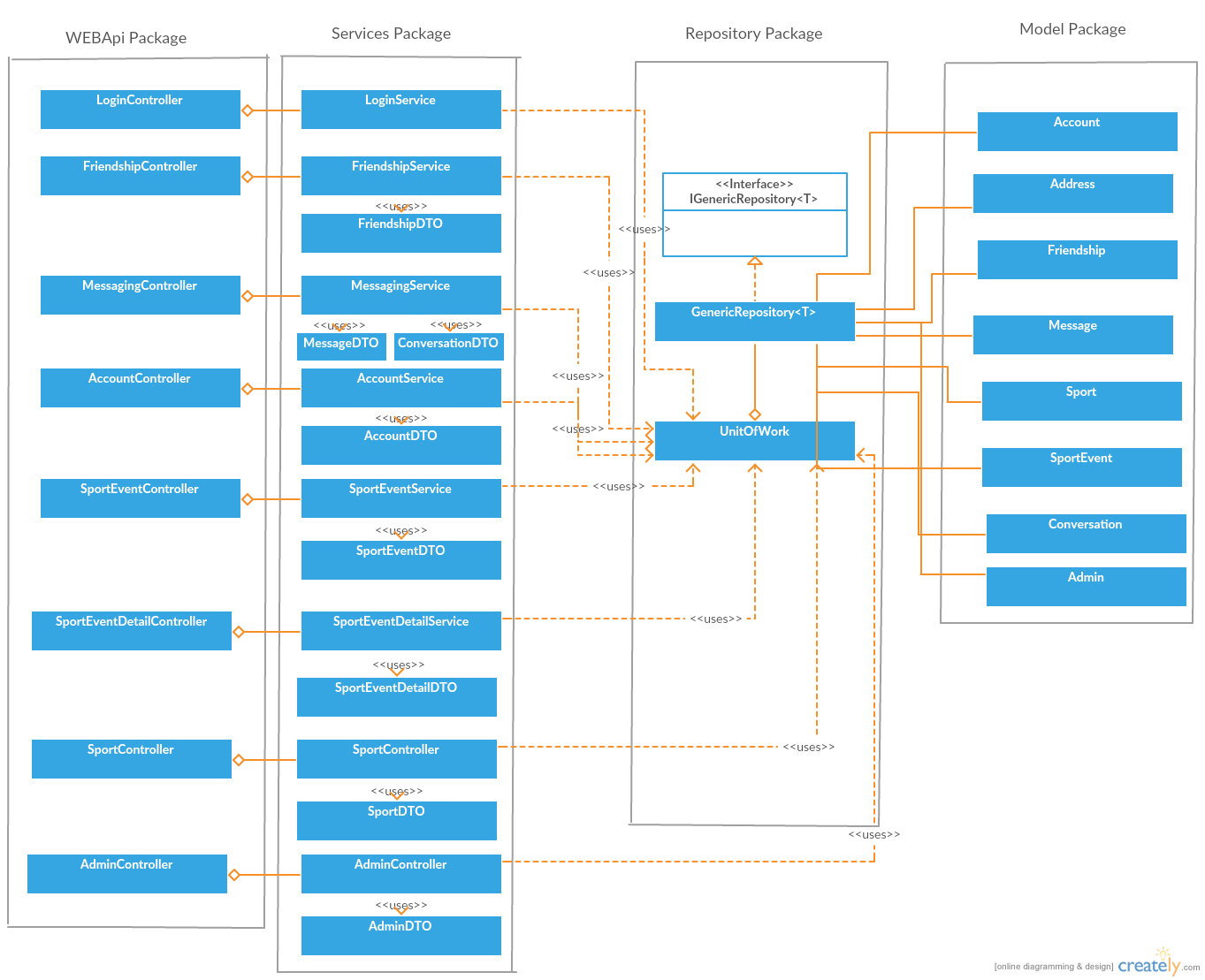
Sequence diagram for creating a sport event.



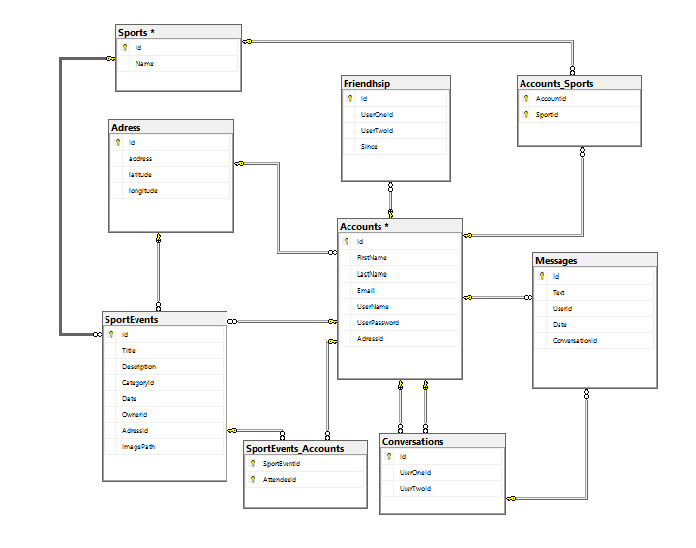
Component diagram for going to a sport event.



**4.7 Class Design**

****

**4.8 Database diagram**

****

**5. Construction and Transition**

**5.1 System Testing**

After all small modules was tested with unit testing, it is time to combine them, ant to test the whole application. It were created 4 classes for test ( AccountsTest, SportEventTest, FriendshipTest and MessengerTest). In all these classes were tested the main functionalities from AccountService, SportEventService, FriendshipService and MessengerService. The method that were tested in AccountsTest are :

1.SignUpAndLogin() – was created a new AccountDTO after that it was added to DB, and the login method from Account service should return the Account , and with assert is verified his name, with the name of user which was added.  
 2. EditAccount() – it is the same as the first method, but after the user was added, the last name and address fields were modified, and was executed an edit method. With assert method is check if the fields were modified.

3. AddAccountWithTheSameUserName() – after was added an user, it is created another user with the same username as the first. The method of addAccount should return null, and user will not be added.

The method that were tested in SportEventTest are :

1.AddSportEvent – a new Sport Event was created and was added in the database, after that it is verified if that user exist in database.

2.UpdateSportEvent – is the same as the first method, but after that the title field was modified and was executed an edit method. With assers is check if title is the new title.

3.GoingSportEvent – after a sport event was created, it was created an user and with method goingSportEvent(sportEventId,userId) the user was added to the attendees of that sport event. After that was checked if that user exist in the list of all users which are goind to that event.

4.NotGoingSportEvent- is the same as goingSportEvent, but after that the notGoingSportEvent is executed and it is checked if that user doesn’t exist in the attendees list.

The method which was tested in FriendshipTest is AddRemoveFriend – Firstly, were created two users which were added in the database. The addFriend method was used from FriendshipService in order to create friendship relationship. With method GetAllFriendship, were optained all friends of user1. With assert it was checked if the user2 is in that list. After that it was removed from friends, and this time it was checked if it doesn’t exist in that list.

The method which was tested in MessengerTest is SendMessage – Firstly , were created two users which were added in the database. Was created a conversation between them and after that a message from user1 with the conversationId. The message was posted in the DB, and after that was optained all message from their conversation, and was verified if that message exist in the list of all messages.

**5.2 Future improvements**

1. Better design
2. Structure and implementation improvements
3. Admin can add sport news
4. Users can add comments and give a note to sport events.
5. Users can share events to/from facebook
6. Encrypted password
7. Profile image for users
8. Share events, news on friends landing page

**6. Bibliograpghy**

Entity Framework - <http://www.entityframeworktutorial.net/code-first/entity-framework-code-first.aspx>

Web API - <https://www.asp.net/web-api>, https://docs.microsoft.com/en-us/aspnet/web-api/

Angular 2 JS - <https://angular.io> , <http://www.angular2.com/> , <https://www.tutorialspoint.com/angular2/>

Design Patterns - <http://www.dofactory.com/net/design-patterns>, <https://sourcemaking.com/design_patterns>

SQL Database - <https://www.w3schools.com/sql/sql_intro.asp>

Designing Web Applications - <https://msdn.microsoft.com/en-us/library/ee658099.aspx>