Technical University of Moldova
Inginerical department S.A.

Report

№: 4

AMOO

Subject: Collaboration diagrams

Author: Terman Emil FAF161
Prof: R. Melnic

Chisinau 2018

1 Theory

1.1 Collaboration diagrams

Unlike sequence diagrams, the collaborations has no indication of the interaction time of the objects. This kind of diagram is more focused to represent WHAT are the interactions between objects than when.

1.2 Design principles

All the following concepts represent basic analysis and design techniques. They are interrelated and normally used together during software development. We use them even though we are not always aware of it. Deeper understanding of these concepts helps us to be more accurate and effective.

1. **Abstraction** - in general, it's the process of consciously ignoring some aspects of a subject under analysis in order to better understand other aspects of it. In other words, it is some kind of a simplification of a subject. In software in particular, analysis and design are all about abstraction.
   - when the architecture is modeled, you concentrate on high-level modules and their relationships and ignore their internal structure;
   - each UML diagram gives a special, limited view on the system. Therefore, it focuses on a single aspect and ignores all other things (sequences abstract objects and messages, deployment abstracts network and servers, use cases abstract system users and their interactions with a system, etc);
   - so, abstraction is used to generalize objects into one category in the design phase. For example in a travel management system you can use Vehicle as an abstract object or entity that generalizes how you travel from one place to another;
2. **Encapsulation** - refers to the process of an object controlling outside access to its internal data. This also means to hide functions and methods of a class;
3. **Generalization** - is a relationship in which one model element (the child) is based on another model element (the parent). Generalization relationships are used in classes, components, deployments and use-case diagrams to indicate that the child receives all of the attributes, operations and relationships that are defined in the parent. This also doesn't have names.
4. **Decomposition** - is an application of the old good principle "divide and conquer" in software development. It is a technique of classifying, structuring and grouping complex elements in order to form more atomic ones, organized in a certain fashion and easier to manage. In all phases there are lots of examples:
   - functional decomposition of a complex process to hierarchical structure of smaller sub-processes and activities;
   - high-level structure of an complex application to 3 tiers - UI, logic and data;
   - UML packages are a direct use of decomposition on the model level - use packages to organize your model;
   - to have a good level of understanding of decomposition, you should first understand the concepts of association, composition, and aggregation;

2 Tasks

The following diagrams are the exact copy from the previous laboratory work, but transformed in collaboration diagrams. For a more detailed description, please refer to my previous laboratory work.

- **Figure 1 - Registration**
  1. open reg. page;
  2. show reg. page;
  3. enter reg. data;
  4. validate reg. data;
  5. load terms;
  6. return terms;
  7. show terms;
  8. accept terms;
  9. server validate reg. data;

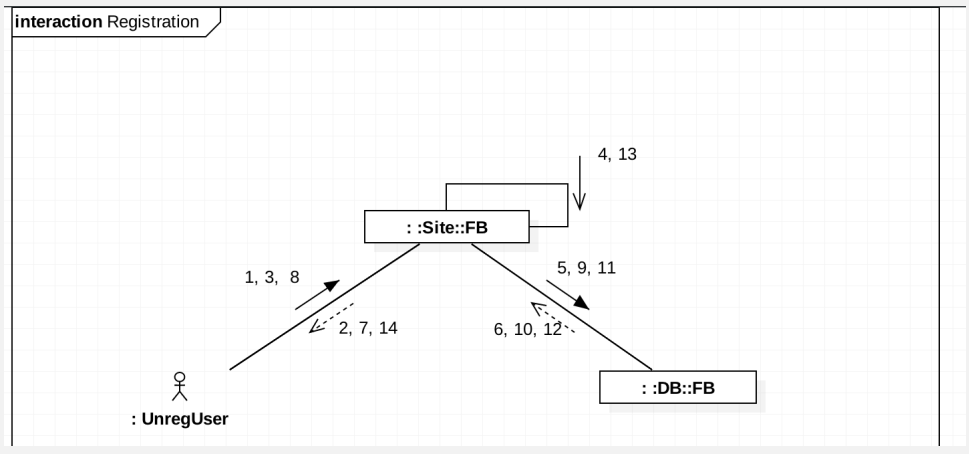
10. valid reg. data;
11. register user;
12. user registered;
13. send email confirmation;
14. confimation sent;

– **Figure 2 - Friend request**

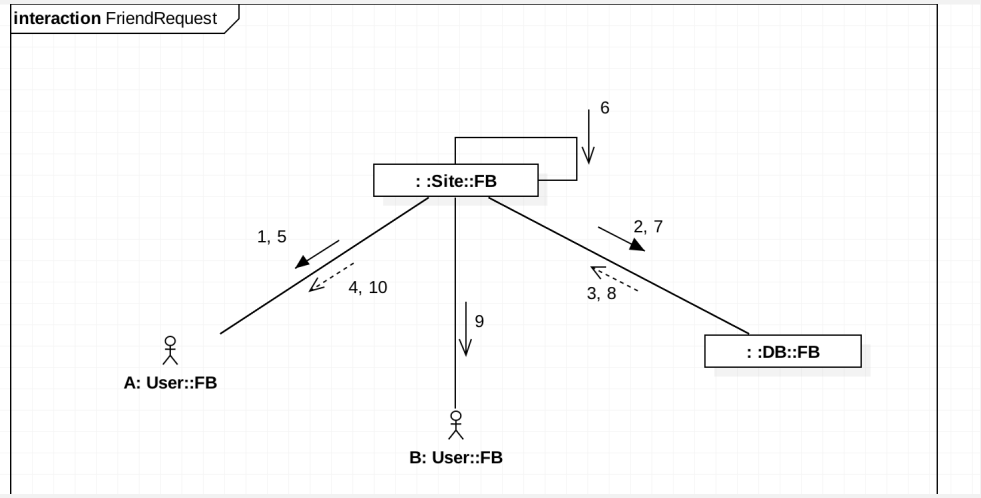
1. access target user;
2. find user;
3. return matching users;
4. show possible users;
5. send friend request;
6. validate request;
7. register friend request;
8. registered friend request;
9. notify user;
10. successful friend request;

– **Figure 3 - Create an event**

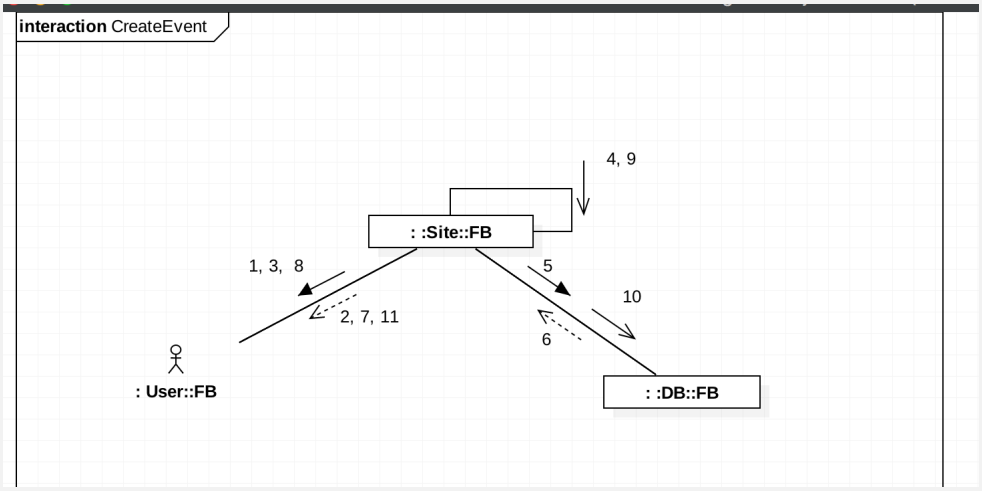
1. access *new event* option;
2. load *new event* page;
3. submit event data;
4. validate request;
5. create event;
6. created event;
7. successful event creation;
8. invite users;
9. validate requests;
10. send invitation requests;
11. invitations sent;



Img 1: Registration



Img 2: Friend request



Img 3: Create event

3 Technologies

Facebook is a social web platform, so I suppose that a user privatness shoul come first. With that said, I will use *Triple DES* or *RSA* encryption algorithms, to ensure everything stays private.

Moving on to profit, like the current facebook, I will sell ads. For that, I would use Google Ads, because they are quite profitable and easy to use.

Since I need to work in a team and continually maintain the web app, I would need a platform where I can easily collaborate with other developers. And **Github** comes with plenty of tools, that fit just right for this problem. Using git, I can create different branches and descriptively commit my changes.

4 Conclusion

This laboratory work helped me better understand the difference between collaboration and sequence diagrams: collab is much more focused on **who** interacts than **when**. Unlike collab, on sequence diagram it's much easier to see the flow of actions. This led me to the conclusion, that if I were to work on big projects, it would be useful to have both collaborative and sequence diagrams, so that the whole team doesn't get conffused with different ideas.