POLITECNICO
MILANO 1863

# Code Kata Battle

A software development platform based on a code kata approach

Francesco Spangaro – Luca Tosetti – Francesco Riccardi

# CKB Platform's Goals

The most important goals we defined are:

- G1: Allow Educators to create new tournaments;
- G2: Allow Educators to create new battles;
- G5: Allow Students to subscribe in tournaments;
- G6: Allow Students to participate in battles.

# CKB Platform's Alloy

The CKB Platform's behavior was modelled using Alloy 6.0. Here are some of the most important facts we wrote to ensure that the CKB Platform would behave in the expected way.

- pred NewGrantedPermit[ t : Tournament, e : Educator, p, pf : Permit ] {

```
//Preconditions
    e != t.creator
    e not in p.battleCreators
    t = p.tournament
//Postconditions
    pf.battleCreators = p.battleCreators + e
    pf.tournamentCreator = p.tournamentCreator
    pf.tournament = p.tournament
}
```

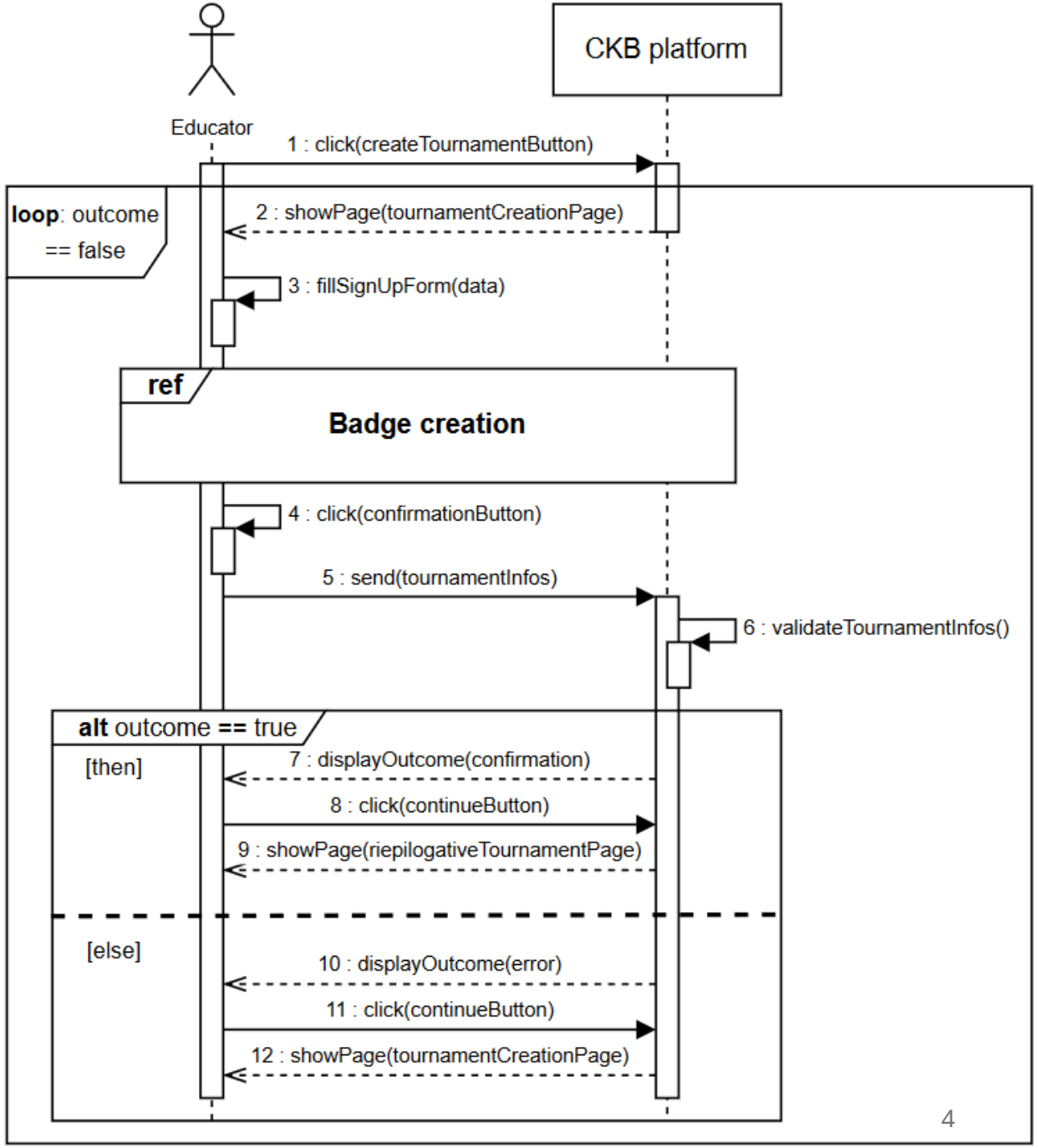
- fact StudentsGroupBattleInTournament {

```
    all g : Group | all s : g.students |
    g.battle.tournament in s.tournament and
    g.battle.tournament in g.groupCreator.tournament
}
```

# CKB Platform's Use Cases - 1

One of the most important part of the CKB Platform is to let educators create new tournaments, containing battles and badges.

This is all explained in great detail in section 3.2.3, use case 3: «Educator creates a new tournament».

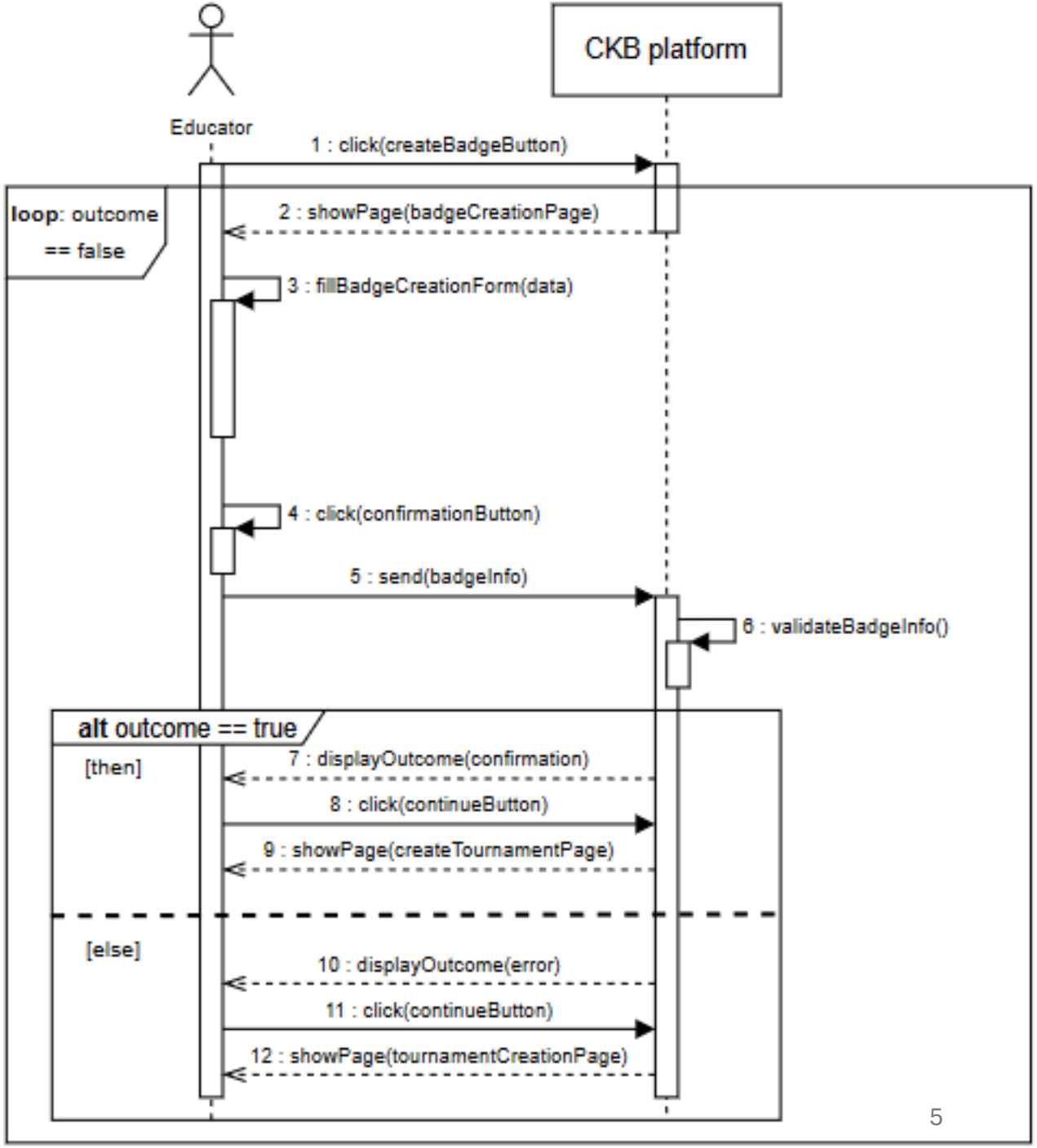


# CKB Platform's Use Cases - 2

The badge use case.

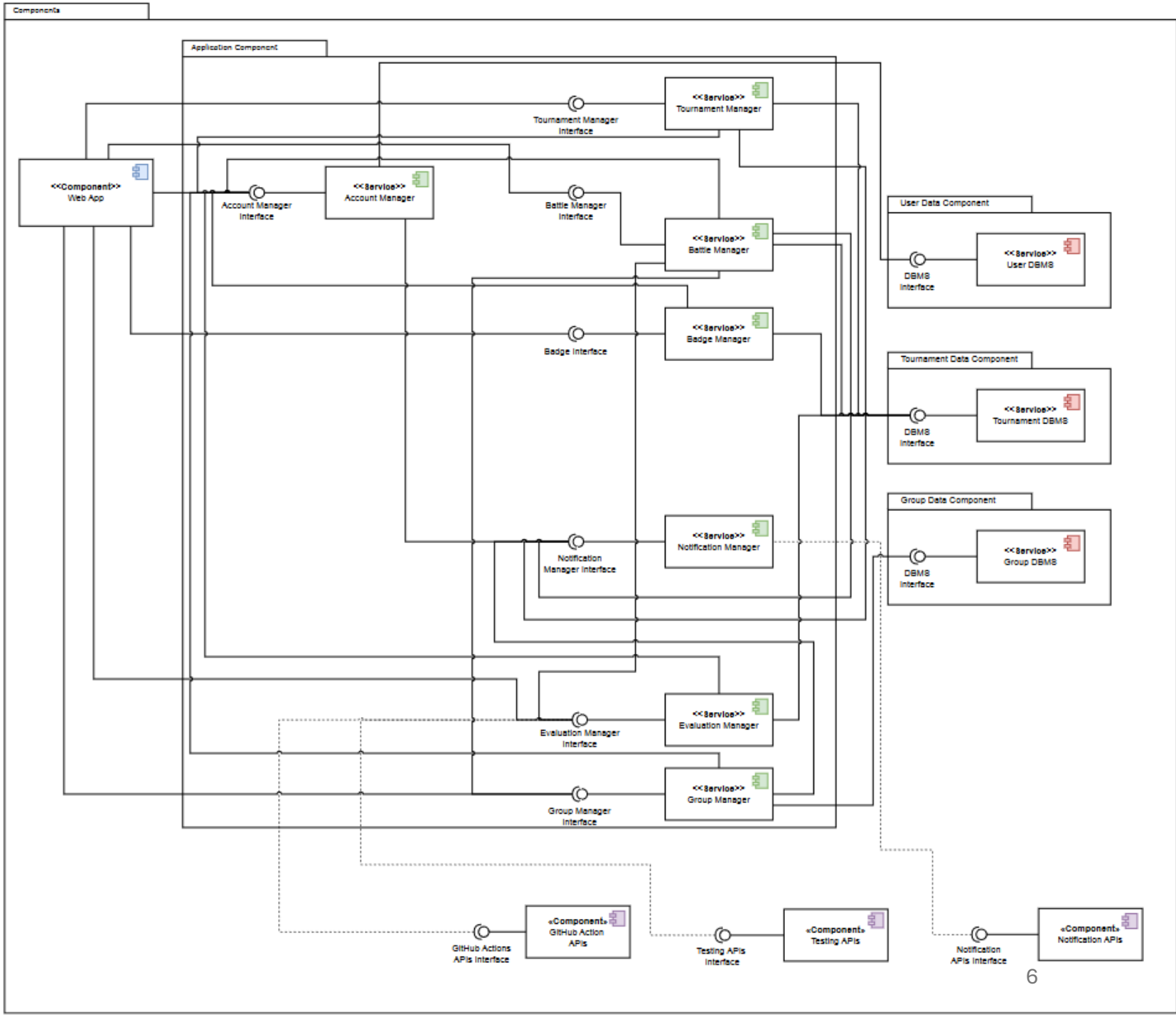
This explains the process of creating one or more badges during the creation of a new tournament.

This is all explained in great detail in section 3.2.3, use case 4: «Educator creates new badges».



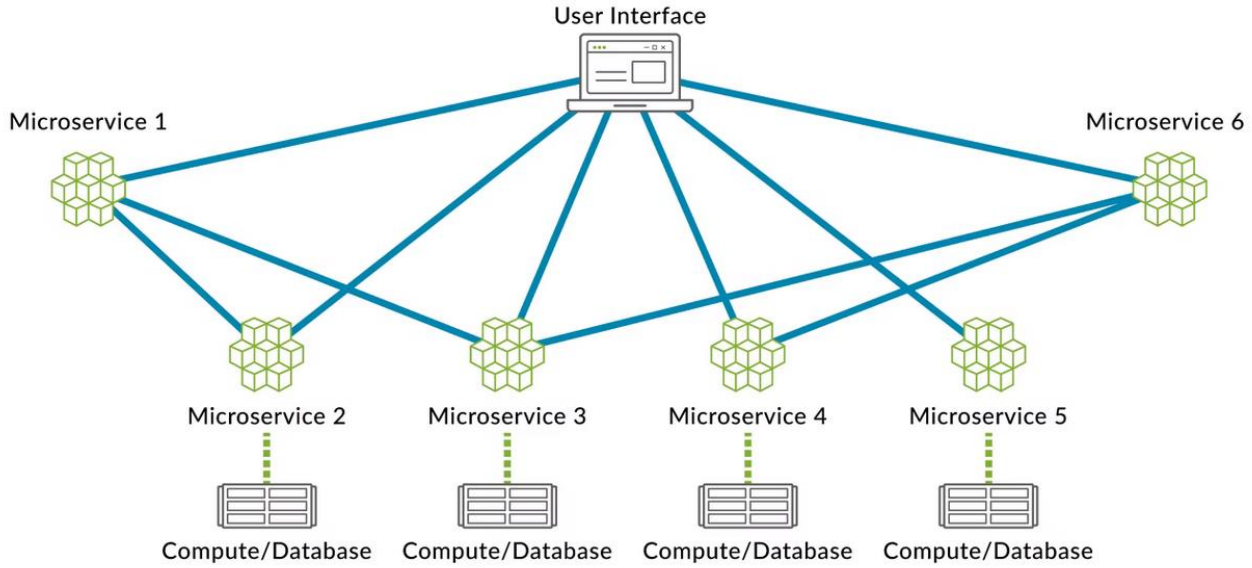
# CKB Platform's Component View

The Component Diagram is what we created to describe, in a more detailed way, all the components needed for a full implementation of the CKB Platform.

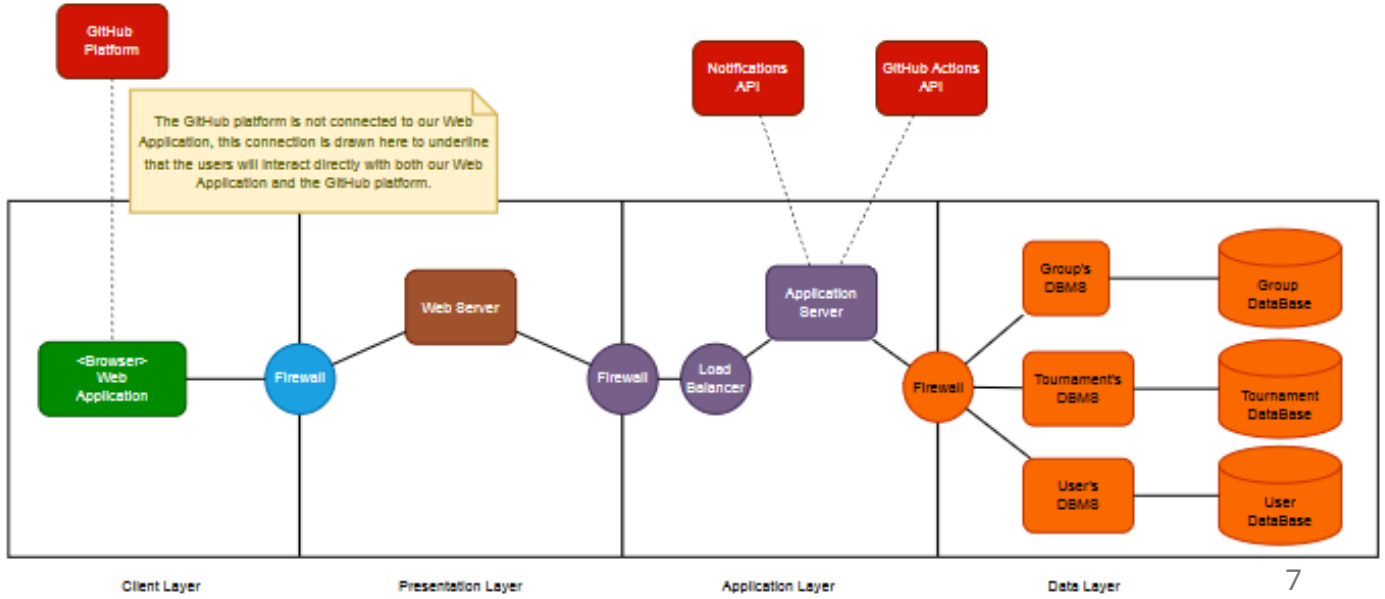


# Selected architectural styles and patterns - 1

- MicroServices architecture:

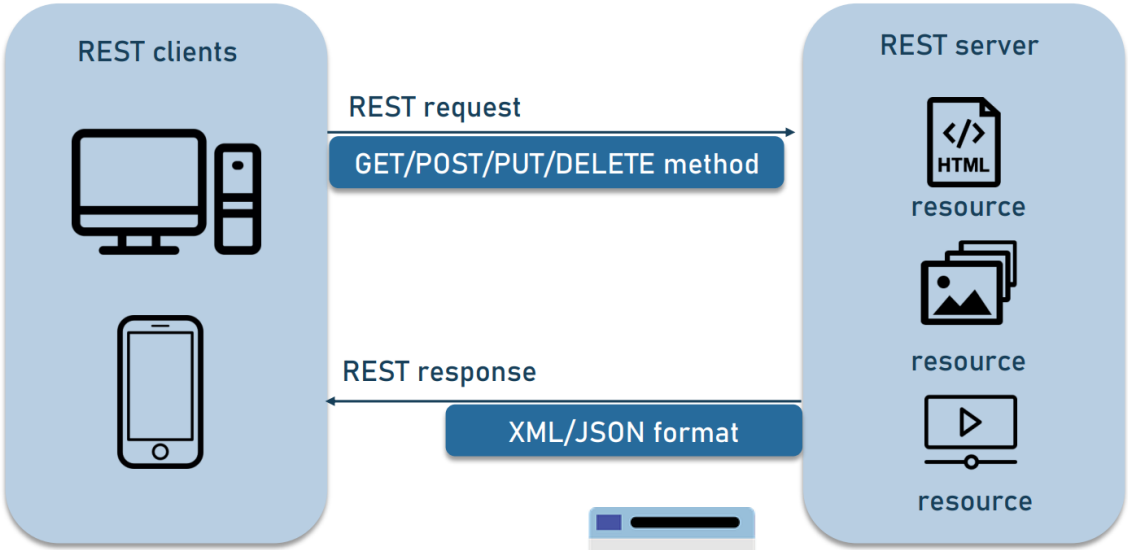



- Four-tier architecture:

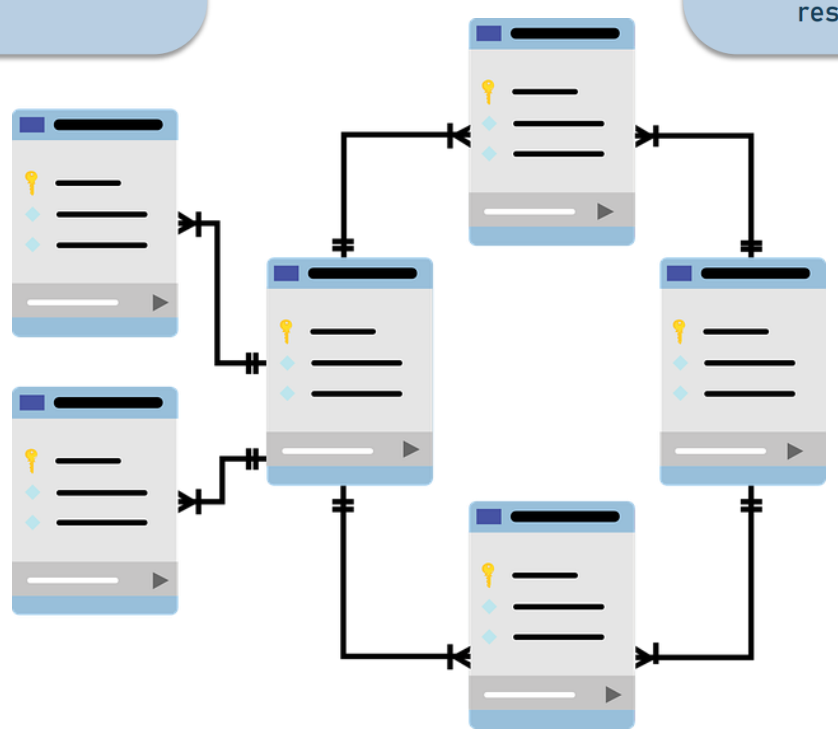


# Selected architectural styles and patterns - 2

- REST architecture:




- Relational Database:



# Implementation and Testing

The implementation and integration plan is divided in:

- Bottom-up;
- Thread.

For testing the most important types we decided to use are:

- Unit testing;
- Functional testing;
- Performance testing.