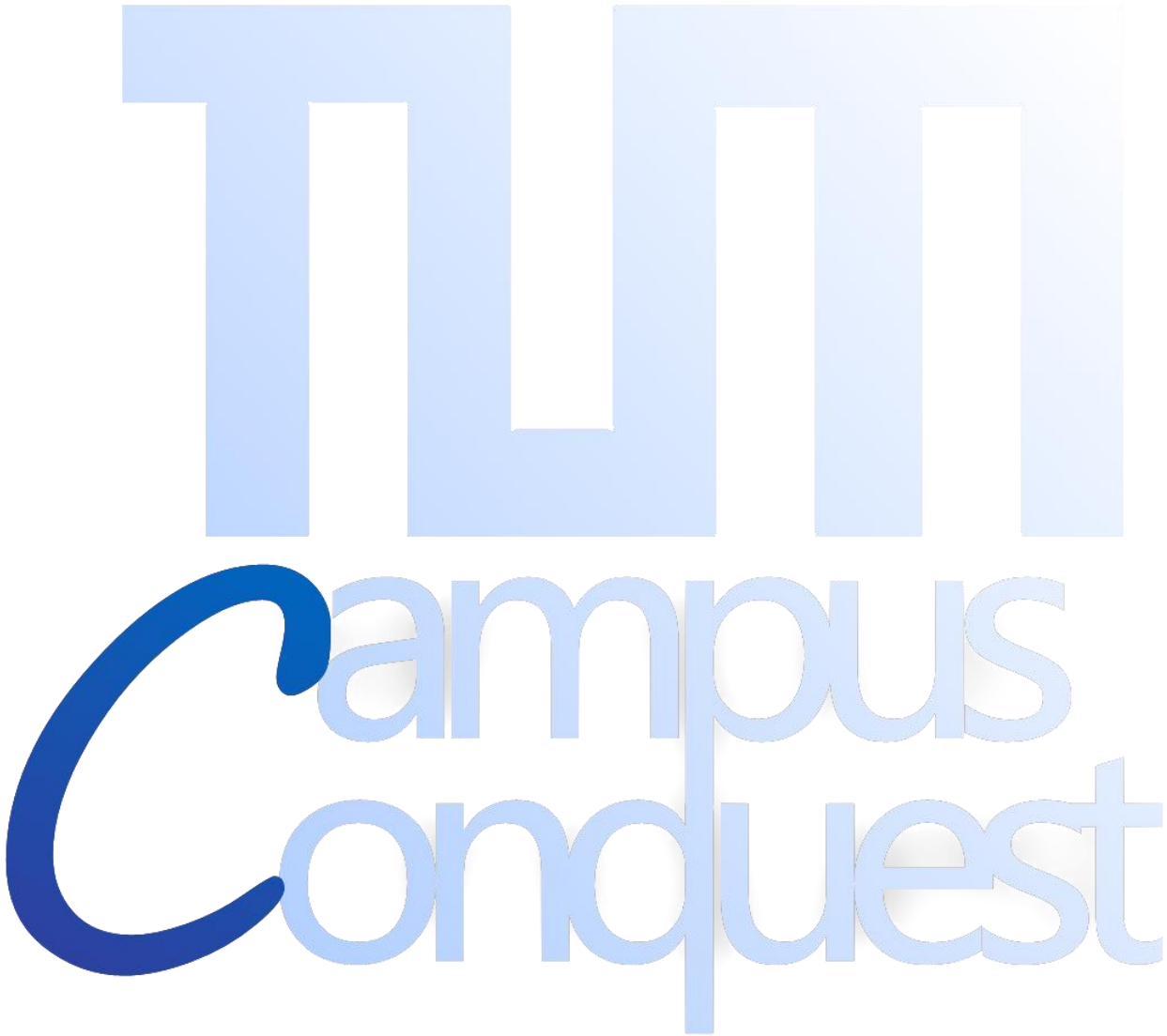
Campus Conquest
Final Presentation

Group 8
Michael Sailer, Paul Preißner, Jonas Mayer,
Benedict Drechsler, Jean-Paul Vieira, Julian Frattini

[IN 0036] Praktikum Social Gaming


Campus
Conquest

Overview

- Game Concept
- Main Game Loop
- Obstacles and Problems in the Development Process
- Working in a Team
- Screencast

Part I

Game Concept

Game concept

Game idea

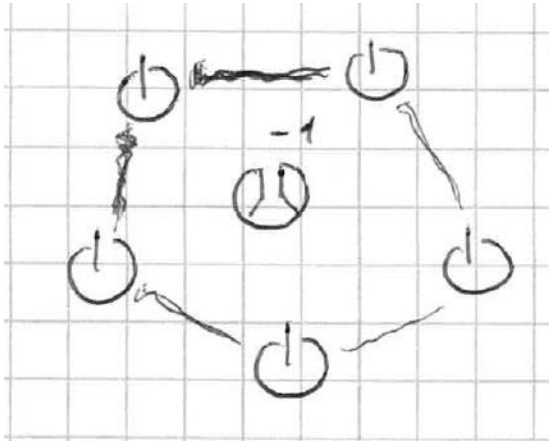
Similar to Conquest game modes from popular RTS/FPS games:

- Capture zones/flags
- To be conquered by factions through players „being there“
- Influenced by player class/skill factors
- Fight enemies through „drawing“ polygons with your mates

→ simple to learn, group dynamic can lead to complex matches/rapid twists

Game concept

Social context



Short term:

- Per-match statistics
- Local group dynamics (+ real team work/conversation)
- „Commander-Mode“/per-match ranks

Long term:

- Rivalry between faculties
- Hierarchy system, faculty-internal ranks
- Member scoreboards
- Overarching interaction/discussion (forums, events, surveys)

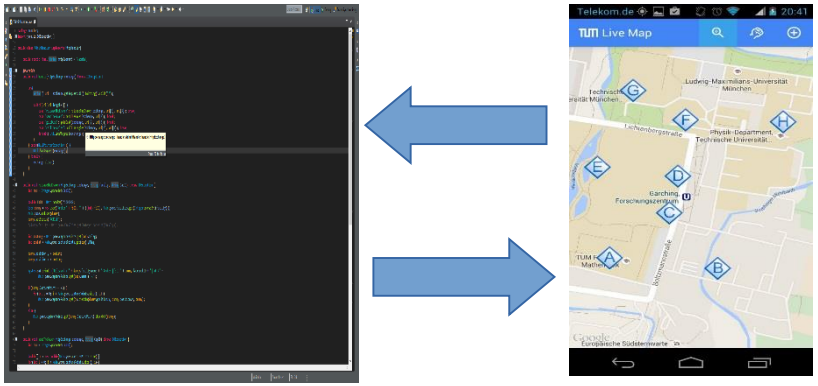
Part II

Main Game Loop

Main Game Loop

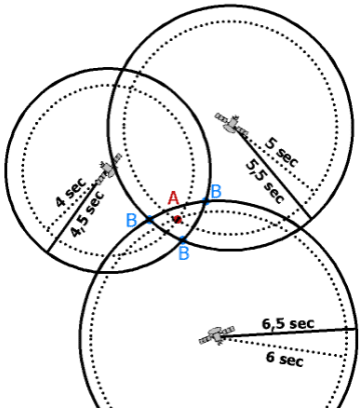
Game Logic

- Client-side
- Server-side
- HTTP



Game Data

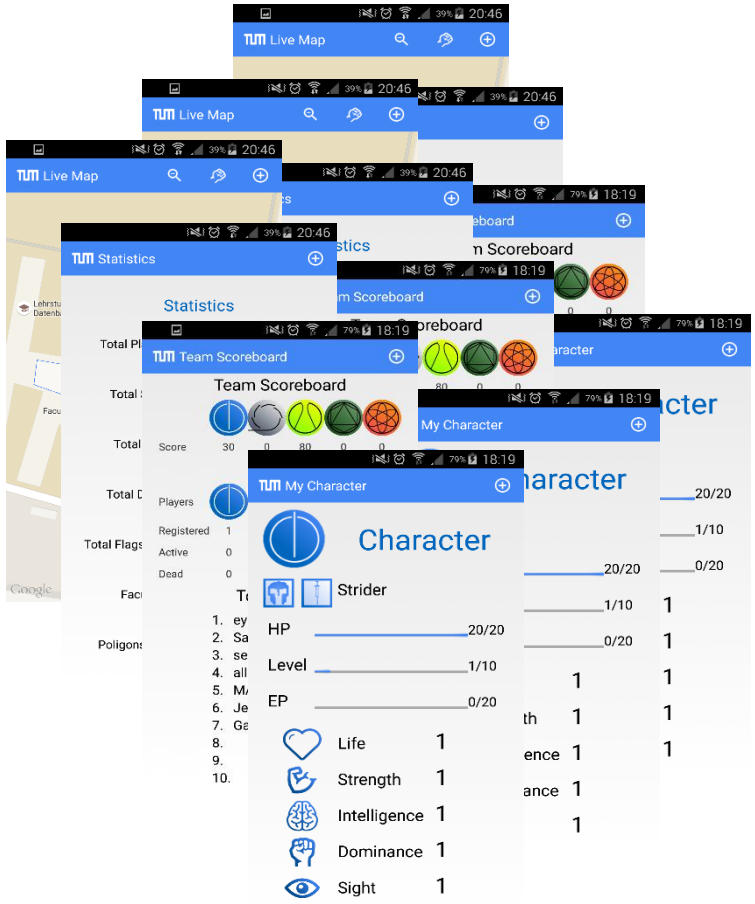
- Simulation Data
- Statistics and Social Context Data



Main Game Loop

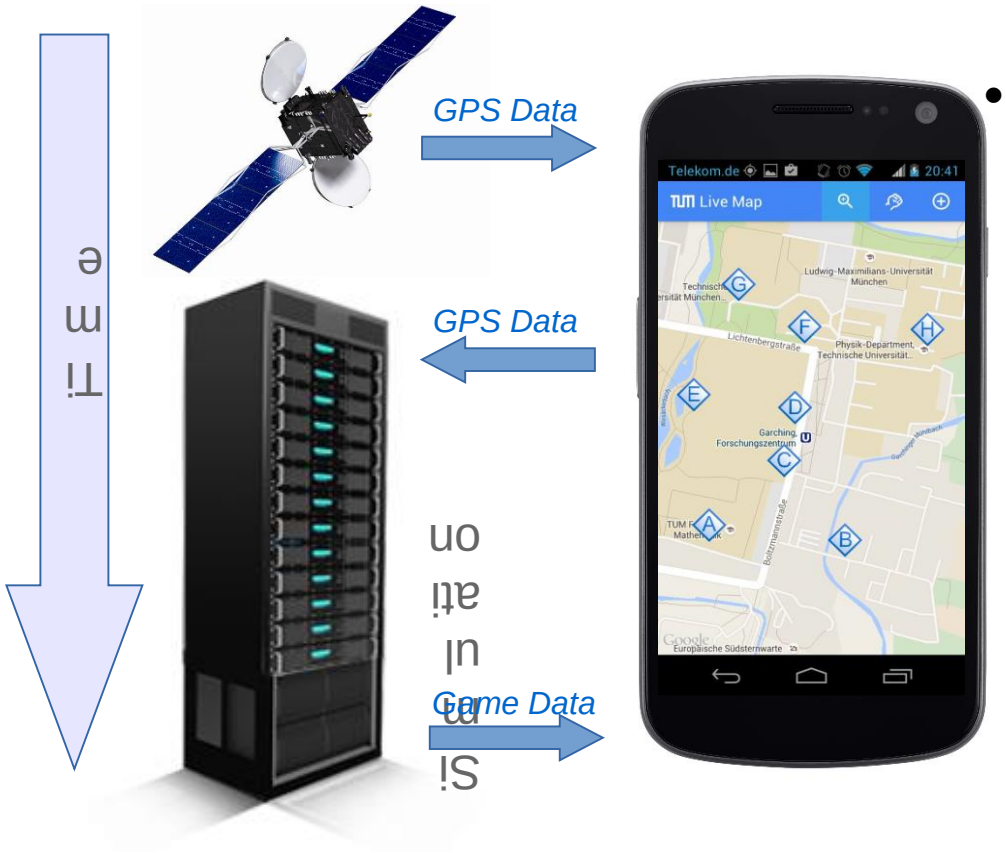
Game Data

- Statistics and Social Context Data:
  - Player Score/Deaths/Kills
  - Faculty Scoreboard
  - Game Statistics
  - Single event data
- Simulation Data:
  - Users' GPS coordinates
  - Users' status (HP, Energy, isAttacking)
  - Capture Points' status
  - Attack meshes



Main Game Loop

Game Logic



- Simulation data flow:
  - Clients gather GPS data and send it to the server via HTTP Post
  - Server receives GPS data from all clients and use it to simulate one “step” of the game.
  - Clients request the updated game in a specific frequency via HTTP Request
  - This information is processed and displayed to the user on the client

Main Game Loop

Game Logic

- Statistics and Social Context
  - A client sends randomly, when the user wants to see this data, a HTTP Request to the server.
  - The server packs all the data on a JSON Object and sends it back to the specific client.
  - The client then processes the information and display it back to the user

Part III

Obstacles and Problems in the Development Process

play Framework

What the Play Framework offers:

- lightweight, stateless, web-friendly architecture
  - –run on request
  - –hot-code reload
  - –requires http-Requests
  - –requires data bank

What we need:

- continuous game world simulation
- efficient support of multiple clients
- fast responsiveness
- continuously running server

Solution:

We‘re writing our own server from scratch

```
public String name;
public String letter;

public Polygon p;

public class Contestants {
    public List<User> contestants;
}
public Contestants[] c;

public double progress = 0.0;
public int dominatingFaculty = -1;
public boolean captured = false;
```

Flag Capturing Process

Problem:

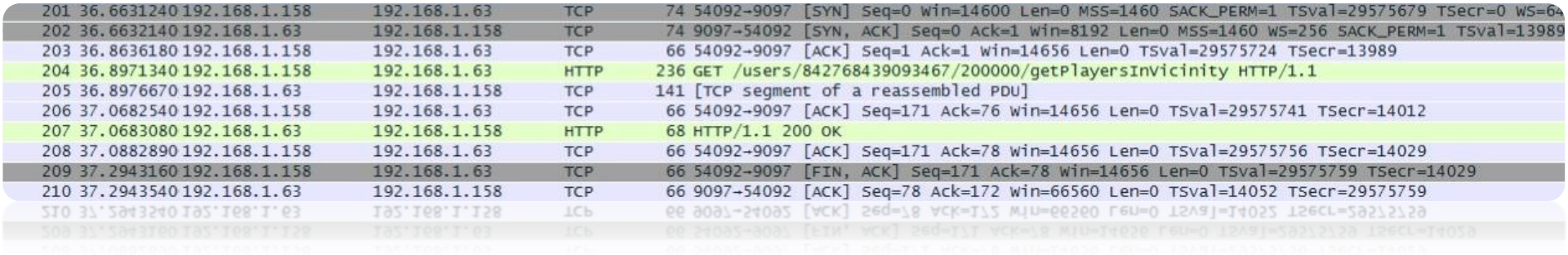
Transfer the known 1:1 concept in a 1:n or n:n context

- multiple D.O.F for solutions
- Created one of the most controversial discussions in our team
- Different concepts:
  - mixed multi faculty capturing
  - destroy with everyone - capture with team
  - Dominant faculty capture

Our Solution:

- Most numerically dominant faculty starts destroying and capturing the point

HTTP-Requests



```
201 36.6631240 192.168.1.158  192.168.1.63   TCP    74 54092→9097 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=29575679 TSecr=0 WS=64
202 36.6632140 192.168.1.63   192.168.1.158  TCP    74 9097→54092 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=13989
203 36.8636180 192.168.1.158  192.168.1.63   TCP    66 54092→9097 [ACK] Seq=1 Ack=1 Win=14656 Len=0 TSval=29575724 TSecr=13989
204 36.8971340 192.168.1.158  192.168.1.63   HTTP  236 GET /users/842768439093467/200000/getPlayersInVicinity HTTP/1.1
205 36.8976670 192.168.1.63   192.168.1.158  TCP   141 [TCP segment of a reassembled PDU]
206 37.0682540 192.168.1.158  192.168.1.63   TCP    66 54092→9097 [ACK] Seq=171 Ack=76 Win=14656 Len=0 TSval=29575741 TSecr=14012
207 37.0683080 192.168.1.63   192.168.1.158  HTTP   68 HTTP/1.1 200 OK
208 37.0882890 192.168.1.158  192.168.1.63   TCP    66 54092→9097 [ACK] Seq=171 Ack=78 Win=14656 Len=0 TSval=29575756 TSecr=14029
209 37.2943160 192.168.1.158  192.168.1.63   TCP    66 54092→9097 [FIN, ACK] Seq=171 Ack=78 Win=14656 Len=0 TSval=29575759 TSecr=14029
210 37.2943540 192.168.1.63   192.168.1.158  TCP    66 9097→54092 [ACK] Seq=78 Ack=172 Win=66560 Len=0 TSval=14052 TSecr=29575759
```

Goal:

- low latency, high precision networking for best user experience

Problem:

- http-Request rather slow
- 10 messages needed to transport 1 packet of data
- Ok and Necessary for Play Framework
- performance bottleneck in our current game

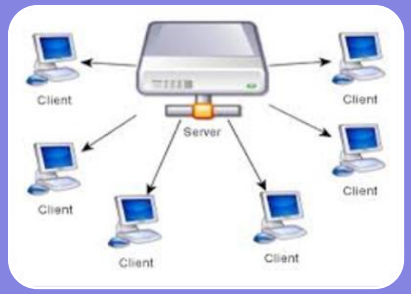
Possible solution:

- Sockets or simply anything but http-Request

Part IV

Working in a Team

Team



Client

Jonas
Paul
Michael

Team

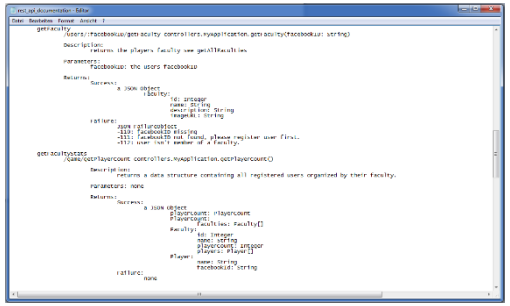
regular
meetings

Server

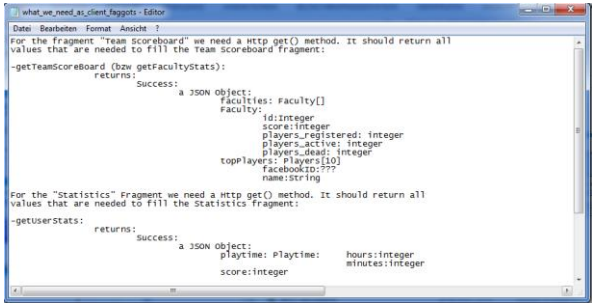
Julian
Jean
Benedict

API Documentation

First:



Documentation on server side



Requests from client side

Later:

DataHandler

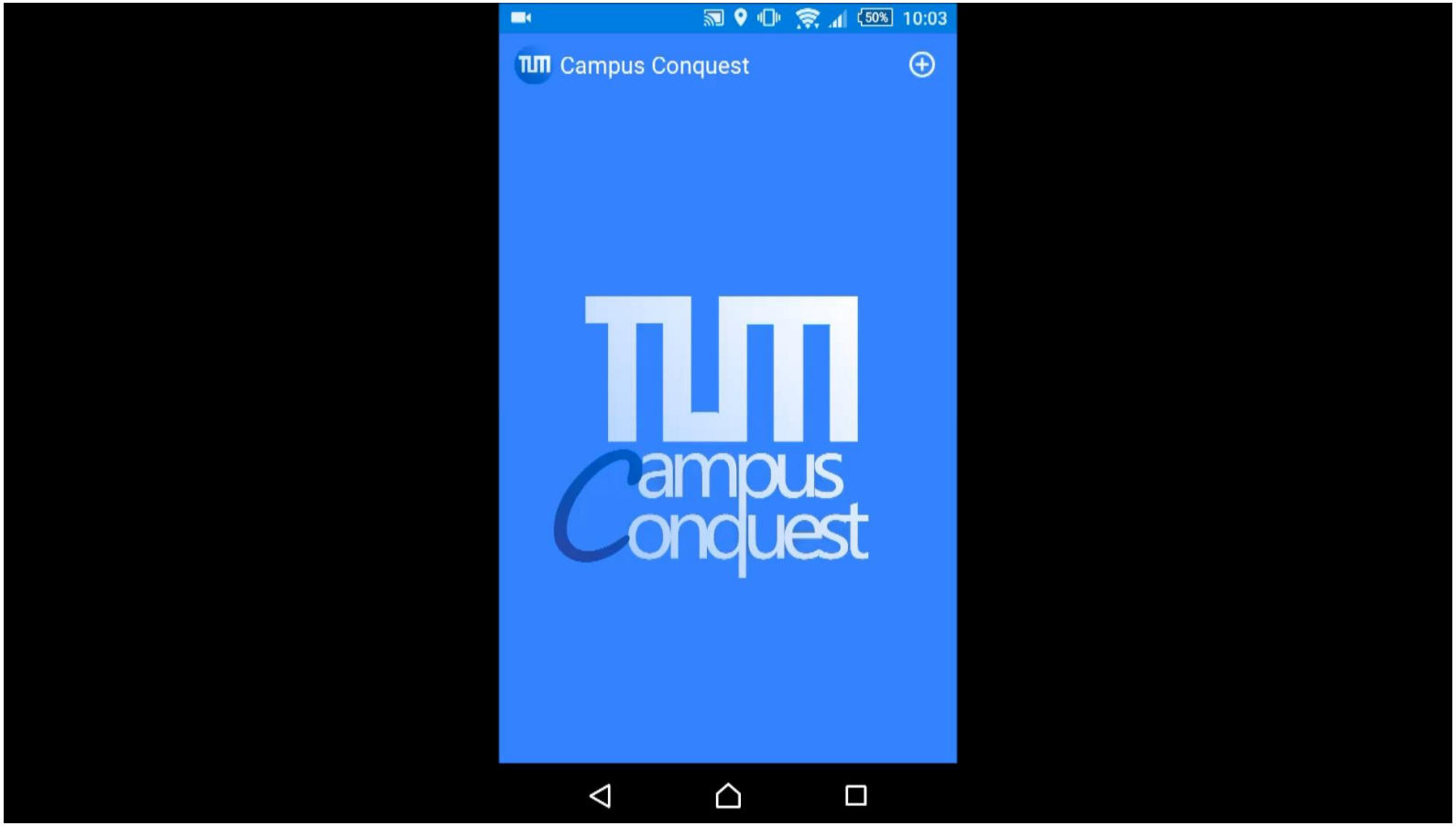
faculties	
Status	verified
Call	/data/faculties
Description	returns a datastructure containing all faculties with a name and. Contains an id for further calls like become a member of faculty X.
Parameters	*none*
Returns	
Success	faculties: Faculty[] Faculty: id: Integer name: String description: String
Fail	*none*

- verified
- implemented
- buggy
- missing
- ...

Part V

Short Screencast

Screencast



Thank you
For your
Attention