Prototype for Continuous Experimentation Server

Janika Kääriäinen
Empirical Software Engineering (ESE)
Supervisors: Tomi Männistö & Fabian Fagerholm

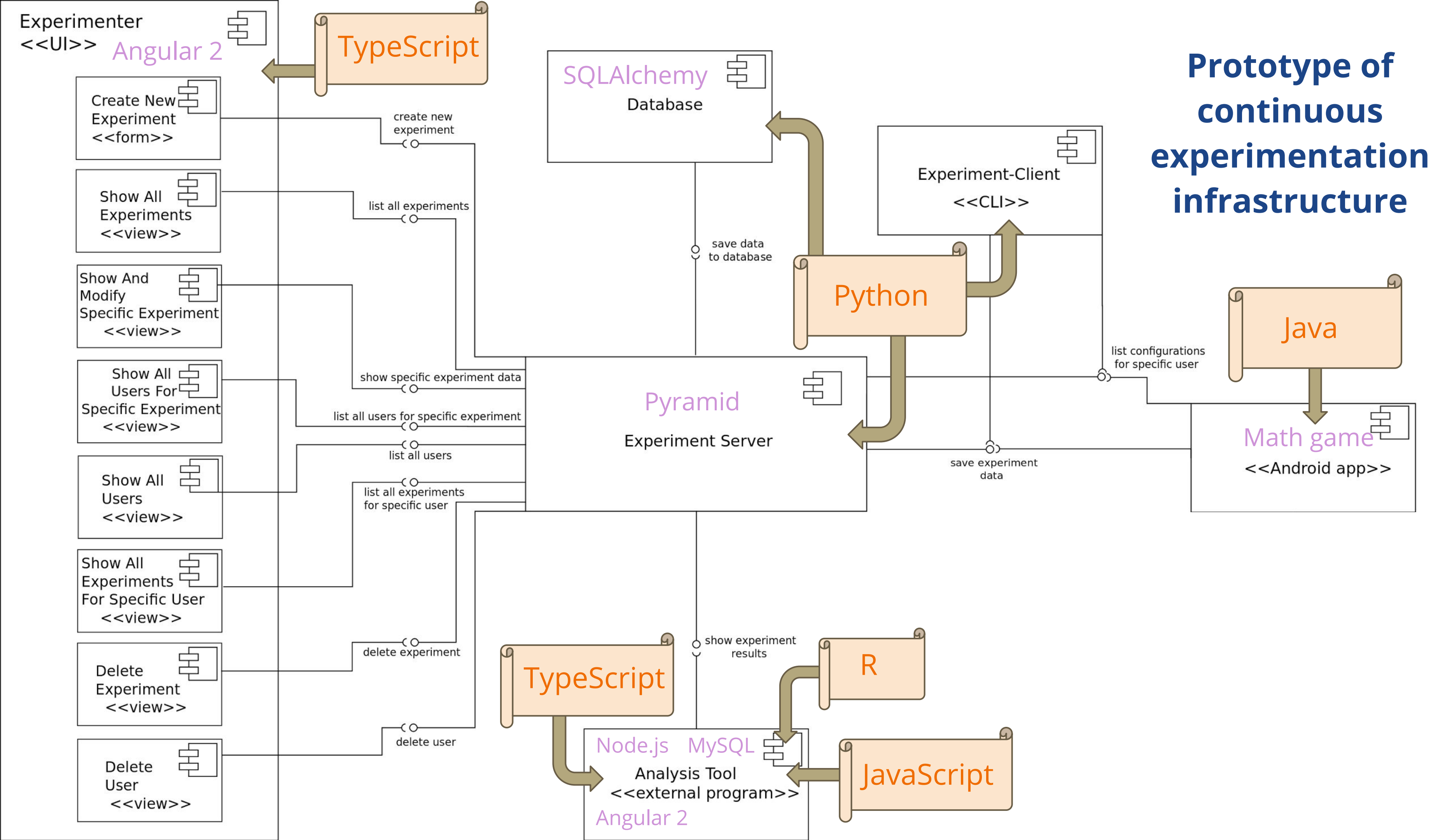
Background

- **Continuous experimentation** is a systematic development approach for linking the customer behavior to the development process.
  - In addition to collecting data from the product or service usage, one proactively introduces changes to the product or service as experiments in order to learn how the customer reacts to them, possibly changing the customer's behavior.
  - Collecting data on these actual changes in the usage allows informed decision making.
  - Requires having a short feedback loop (or rapid customer feedback) in which a product or feature is deployed continuously in order to get feedback quickly from users.
    [Continuous Experimentation Cookbook, in development at CS Department]

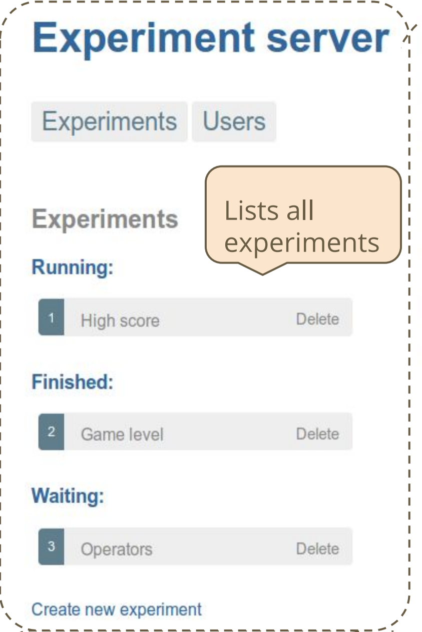
- **The goal** was to implement a prototype server back-end and other infrastructure for continuous experimentation in order to enable empirical research

- **Concepts** (in this case)
  - *A data item* contains information about the behavior of an user
  - *A configuration* is a key-value pair of settings. An application uses a set of configurations for setting up features (e.g. ON/OFF).

Prototype of continuous experimentation infrastructure



UI for experimenters



Experiment server

Experiments | Users

Experiments

Running:

1 High score Delete

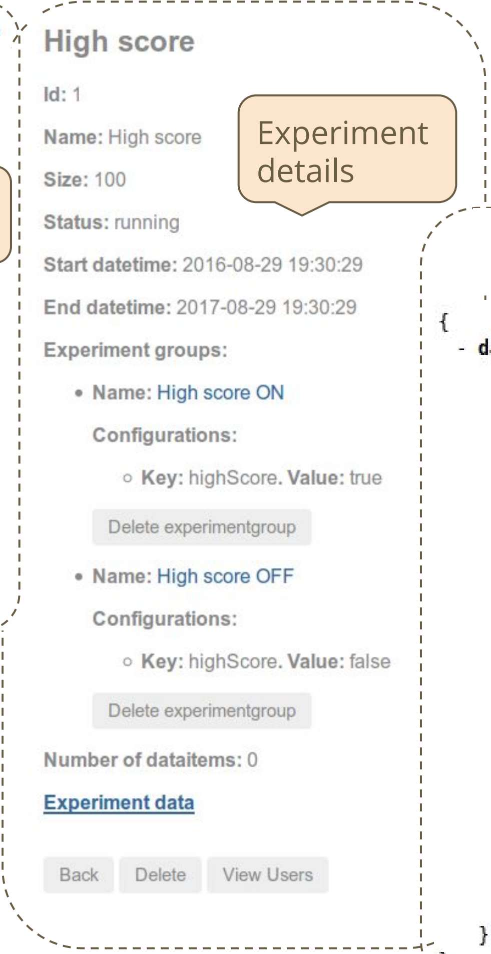
Finished:

2 Game level Delete

Waiting:

3 Operators Delete

Create new experiment



High score

Id: 1

Name: High score

Size: 100

Status: running

Start datetime: 2016-08-29 19:30:29

End datetime: 2017-08-29 19:30:29

Experiment groups:

- **Name:** High score ON

  Configurations:

  - **Key:** highScore. **Value:** true

  Delete experimentgroup

- **Name:** High score OFF

  Configurations:

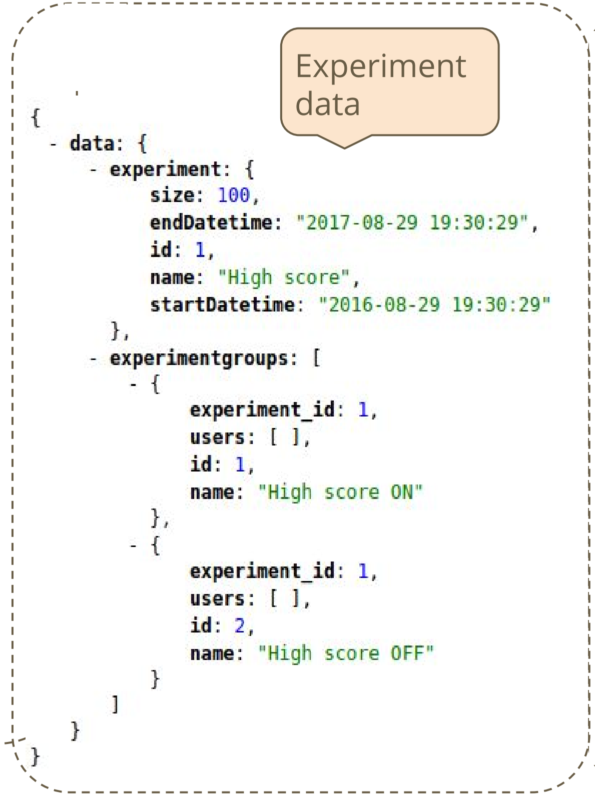
  - **Key:** highScore. **Value:** false

  Delete experimentgroup

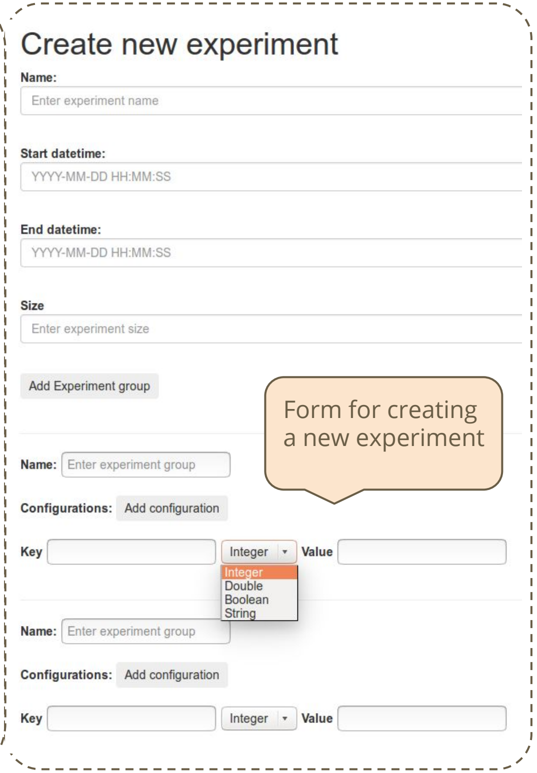
Number of dataitems: 0

Experiment data

Back | Delete | View Users



```
{
 - data: {
    - experiment: {
          size: 100,
          endDatetime: "2017-08-29 19:30:29",
          id: 1,
          name: "High score",
          startDatetime: "2016-08-29 19:30:29"
      },
    - experimentgroups: [
        - {
              experiment_id: 1,
              users: [ ],
              id: 1,
              name: "High score ON"
          },
        - {
              experiment_id: 1,
              users: [ ],
              id: 2,
              name: "High score OFF"
          }
      ]
   }
}
```



Create new experiment

Name: Enter experiment name

Start datetime: YYYY-MM-DD HH:MM:SS

End datetime: YYYY-MM-DD HH:MM:SS

Size Enter experiment size

Add Experiment group

Name: Enter experiment group

Configurations: Add configuration

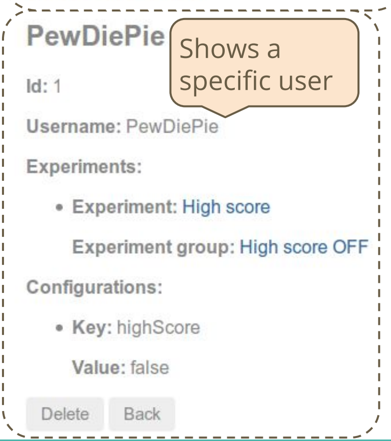
Key | Integer | **Value**

Integer / Double / Boolean / String

Name: Enter experiment group

Configurations: Add configuration

Key | Integer | **Value**



PewDiePie

Id: 1

Username: PewDiePie

Experiments:

- **Experiment:** High score

  Experiment group: High score OFF

Configurations:

- **Key:** highScore

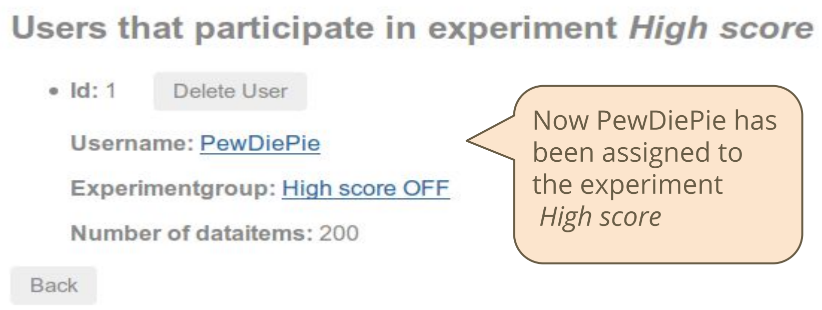
  Value: false

Delete | Back

Command-line client

- **Let's run**

  $ experiment_client.py
  --username=PewDiePie
  --random_dataitems=200
  --key=gameplay
  http://localhost:6543/



Users that participate in experiment *High score*

- Id: 1 Delete User

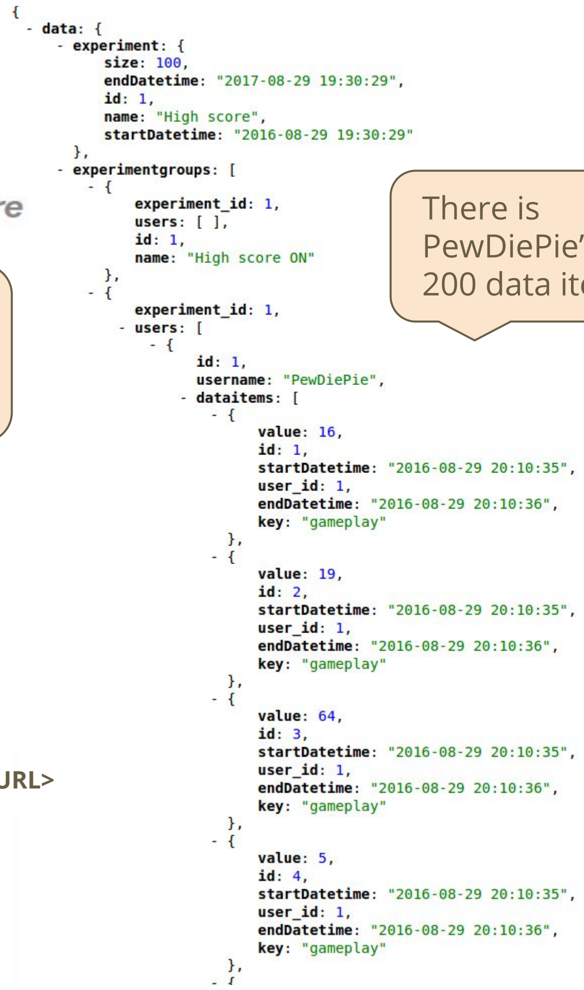
  Username: PewDiePie

  Experimentgroup: High score OFF

  Number of dataitems: 200

Back

Now PewDiePie has been assigned to the experiment *High score*

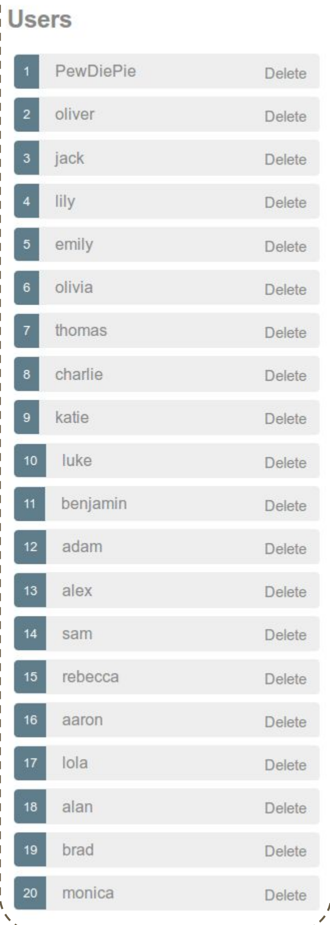


```
{
 - data: {
   - experiment: {
         size: 100,
         endDatetime: "2017-08-29 19:30:29",
         id: 1,
         name: "High score",
         startDatetime: "2016-08-29 19:30:29"
     },
   - experimentgroups: [
       - {
             experiment_id: 1,
             users: [ ],
             id: 1,
             name: "High score ON"
         },
       - {
             experiment_id: 1,
           - users: [
               - {
                     id: 1,
                     username: "PewDiePie",
                   - dataitems: [
                       - {
                             value: 16,
                             id: 1,
                             startDatetime: "2016-08-29 20:10:35",
                             user_id: 1,
                             endDatetime: "2016-08-29 20:10:36",
                             key: "gameplay"
                         },
                       - {
                             value: 19,
                             id: 2,
                             startDatetime: "2016-08-29 20:10:35",
                             user_id: 1,
                             endDatetime: "2016-08-29 20:10:36",
                             key: "gameplay"
                         },
                       - {
                             value: 64,
                             id: 3,
                             startDatetime: "2016-08-29 20:10:35",
                             user_id: 1,
                             endDatetime: "2016-08-29 20:10:36",
                             key: "gameplay"
                         },
                       - {
                             value: 5,
                             id: 4,
                             startDatetime: "2016-08-29 20:10:35",
                             user_id: 1,
                             endDatetime: "2016-08-29 20:10:36",
                             key: "gameplay"
                         },
                       - {
```

There is PewDiePie's 200 data items

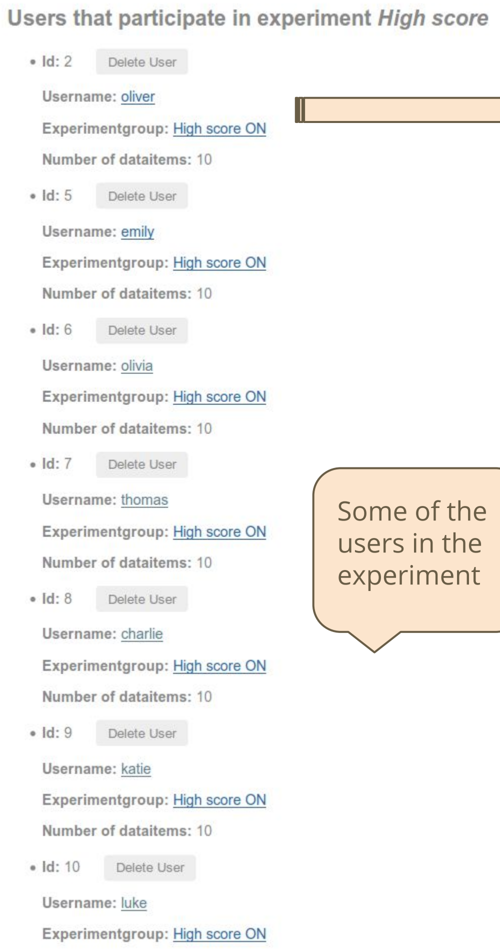
- **Commands**
  - **$ experiment_client.py <URL>**
    - Gets the configuration using the current username in terminal
  - **$ experiment_client.py --username=<username> <URL>**
    - Gets the configuration using the given username
  - **$ experiment_client.py --dataitem=key:value [--dataitem=key:value ...] <URL>**
    - Sends one or several data items
  - **$ experiment_client.py --dataitems=<filename> [--dataitems=<filename> ...] <URL>**
    - Sends several data items, reading the items from file or files. The file must have one key and value per line, separated by a colon (key:value)
  - **$ experiment_client.py --random_dataitems=<n> [--random_min=0] [--random_max=100] --key=<key> <URL>**
    - Send a specified number (<n>) of random values for a specified key. The range of random numbers defaults to [0...100]

Let's simulate more users



Users

1	PewDiePie	Delete
2	oliver	Delete
3	jack	Delete
4	lily	Delete
5	emily	Delete
6	olivia	Delete
7	thomas	Delete
8	charlie	Delete
9	katie	Delete
10	luke	Delete
11	benjamin	Delete
12	adam	Delete
13	alex	Delete
14	sam	Delete
15	rebecca	Delete
16	aaron	Delete
17	lola	Delete
18	alan	Delete
19	brad	Delete
20	monica	Delete



Users that participate in experiment *High score*

- **Id:** 2 Delete User

  Username: oliver

  Experimentgroup: High score ON

  Number of dataitems: 10
- **Id:** 5 Delete User

  Username: emily

  Experimentgroup: High score ON

  Number of dataitems: 10
- **Id:** 6 Delete User

  Username: olivia

  Experimentgroup: High score ON

  Number of dataitems: 10
- **Id:** 7 Delete User

  Username: thomas

  Experimentgroup: High score ON

  Number of dataitems: 10
- **Id:** 8 Delete User

  Username: charlie

  Experimentgroup: High score ON

  Number of dataitems: 10
- **Id:** 9 Delete User

  Username: katie

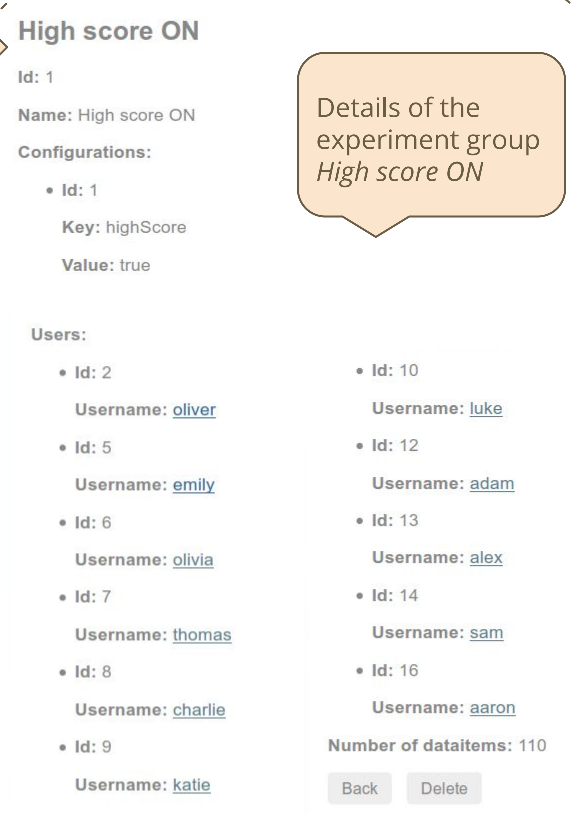
  Experimentgroup: High score ON

  Number of dataitems: 10
- **Id:** 10 Delete User

  Username: luke

  Experimentgroup: High score ON

Some of the users in the experiment



High score ON

Id: 1

Name: High score ON

Configurations:

- **Id:** 1

  Key: highScore

  Value: true

Users:

- **Id:** 2

  Username: oliver
- **Id:** 5

  Username: emily
- **Id:** 6

  Username: olivia
- **Id:** 7

  Username: thomas
- **Id:** 8

  Username: charlie
- **Id:** 9

  Username: katie
- **Id:** 10

  Username: luke
- **Id:** 12

  Username: adam
- **Id:** 13

  Username: alex
- **Id:** 14

  Username: sam
- **Id:** 16

  Username: aaron

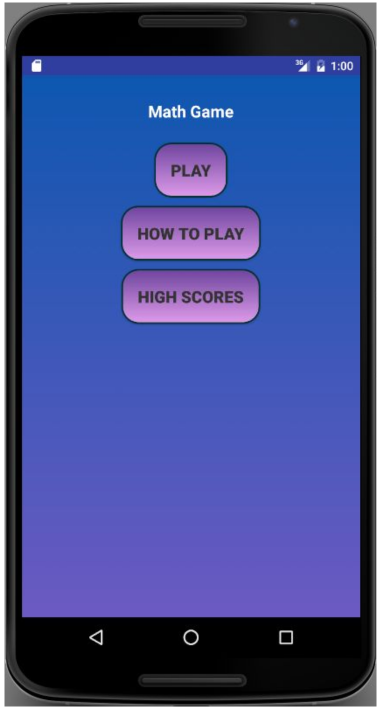
Number of dataitems: 110

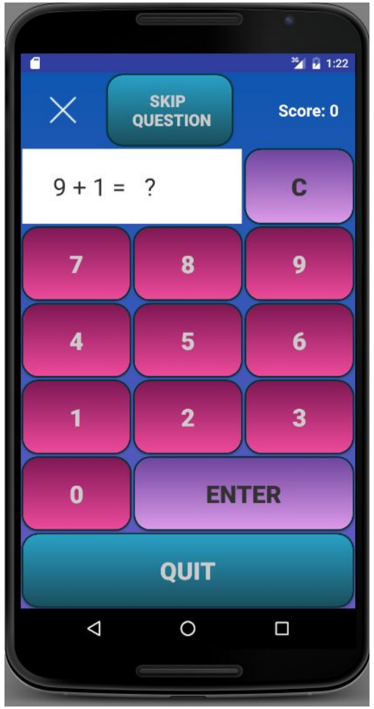
Back Delete

Details of the experiment group *High score ON*

Example: Math game

- Assumption: High score -feature causes more playing
- A/B testing:
  - Group A: High score ON
  - Group B: High score OFF
  - Other features are the same on both experiment groups
    - Game difficulty level (1-5), operators (+, -, /, *, ^, %), how-to guide, possibility to skip a question
- Playing is measured by number of sessions during experimentation time (e.g. two weeks)
- Feature is feasible if playing in group A is significantly higher than in group B
- Groups are equal sizes, users are randomly assigned
- Game has 1000 users

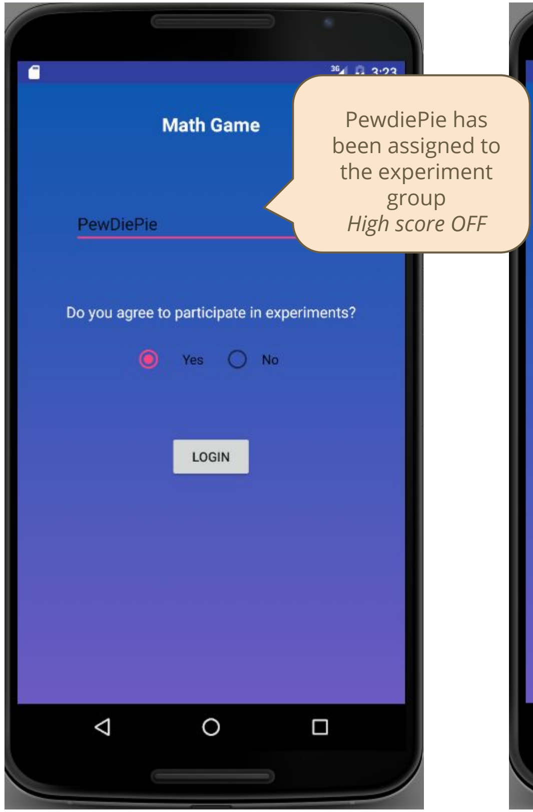


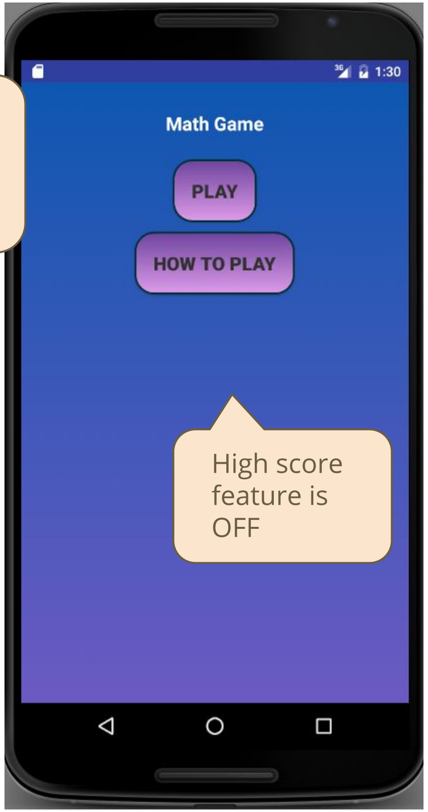


- User login with username

- The user either agrees to participate in experiments or he does not
    - If does, the application sends a request and gets configurations for the user based on the experiment assignments
    - If does not, the application uses the basic configurations and the user is not part of any experiments

- The application posts to the experiment server a data item after each game
    - Key=gameplay, value=1
    - Only if the user agreed to participate in experiments


Math Game
PewDiePie
Do you agree to participate in experiments?
Yes No
LOGIN
PewdiePie has been assigned to the experiment group *High score OFF*

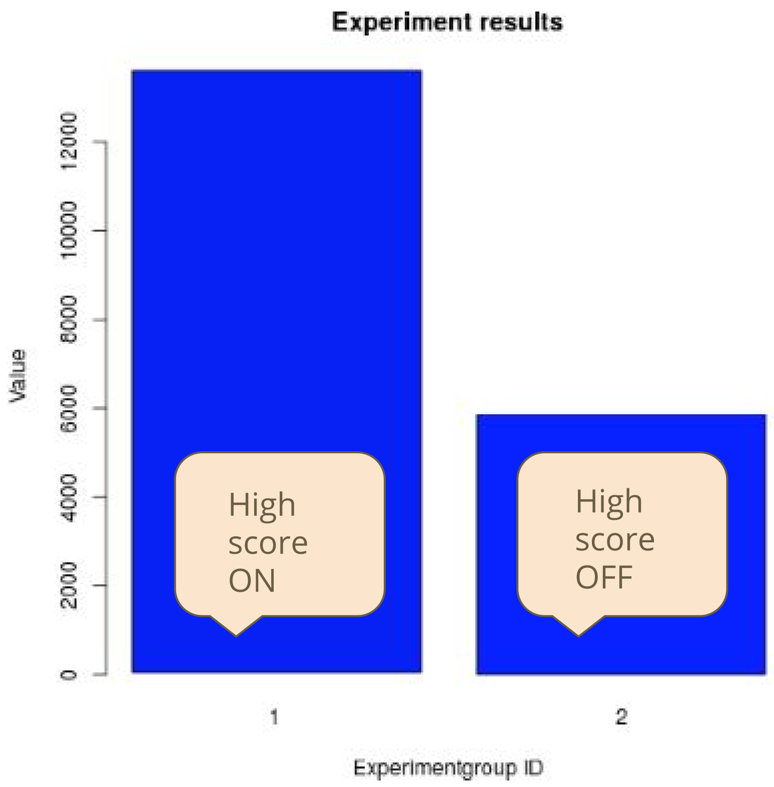


Results from analysis tool

- Analysis tool is a component that fetches data from an experiment, calculates and visualizes results.

- For the simplest JSON form the tool gives an R script for creating a plot from the results.
- The plot has been defined by calculating the sum of each user's value in each experiment group

- Results:
  - Values (gameplays) in group *High score ON* is over double that of *High score OFF*
- Conclusion: high score is feasible



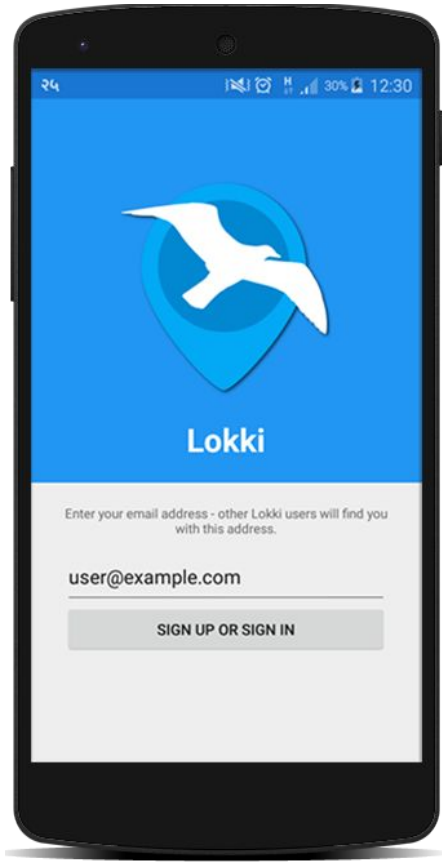
In the future

- The project produced knowledge about how to design infrastructure for continuous experimentation
- The prototype can be used for empirical trials
- Fix known bugs: form validation, workflow etc.
- Increase test coverage:
    - only the server has been tested
    - improve error handling
- Documentation
    - the server is thoroughly documented: component diagram, database diagrams, activity diagrams, sequence diagrams
    - documentation for other components is missing
- Refine the database model?
    - e.g. the *value* attribute of *data item* is integer. It could be also string, boolean, double…
- Making experiments, users (why not everything) more scalable

- **Only sky is the limit**

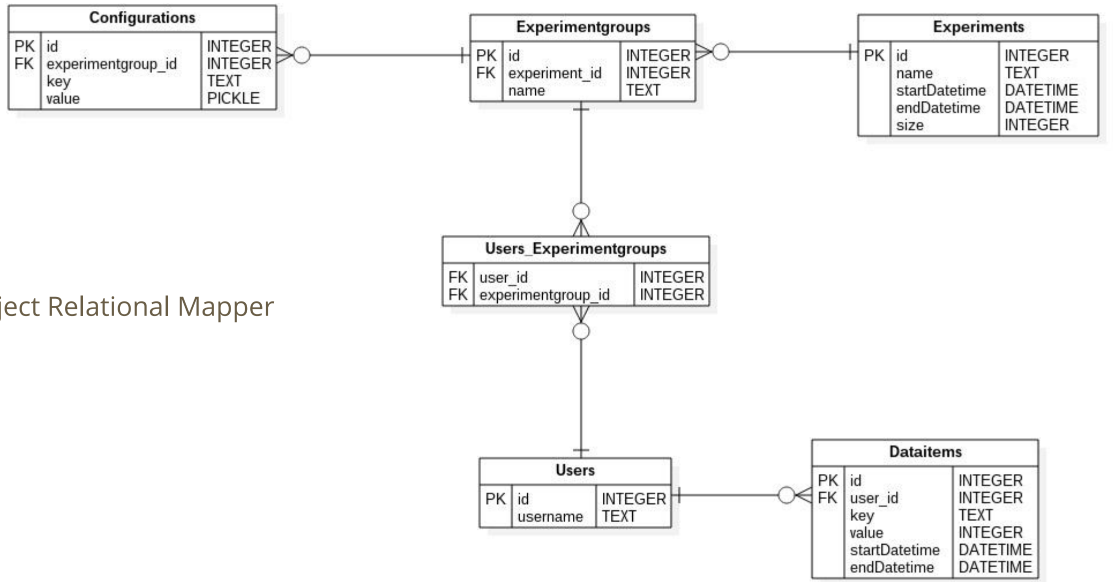
Experiment application

- The intention was to use a real application for testing the infrastructure
  - **Lokki** is an internal startup by F-Secure. It is a free mobile location sharing app for families and other closed groups.
  - F-Secure decided to open source the Lokki source code
  - Since then, students in Software Factory projects have been developed Lokki
- Lokki was not yet ready to be integrated to the experiment server so I created a math game for Android (using some tutorial).


12:30
Lokki
Enter your email address - other Lokki users will find you with this address.
user@example.com
SIGN UP OR SIGN IN

Server

- Backend for the infrastructure
- REST interface
- Database
  - SQLAlchemy is the Python SQL toolkit and Object Relational Mapper
- Tests
  - High level of coverage
  - Unit tests
  - Functional tests (REST interface)



Function	URI	Method	Headers	Access	Payload
create new experiment	/experiments	POST		Experimenter UI	JSON experiment definition
list all experiments	/experiments	GET		Experimenter UI	None
show specific experiment metadata	/experiments/{id}/metadata	GET		Experimenter UI	None
delete experiment	/experiments/{id}	DELETE		Experimenter UI	None
list configurations for specific user	/configurations	GET	User id, client version, ...?	Client	None
list all users	/users	GET		Experimenter UI	None
list all users for specific experiment	/experiments/{id}/users	GET		Experimenter UI	None
list all experiments for specific user	/users/{id}/experiments	GET		Experimenter UI	None
save experiment data	/events	POST	User id, client version, ...?	Client	JSON experiment data
delete user	/users/{id}	DELETE		Experimenter UI	None
show experiment data	/experiments/{id}/data	GET		Analysis Tool	None
delete experimentgroup	/experiments/{expid}/experimentgroups/{expgroupid}	DELETE		Experimenter UI	None
show specific experimentgroup metadata	/experiments/{expid}/experimentgroups/{expgroupid}	GET		Experimenter UI	None
delete user from specific experiment	/experiments/{id}/users/{id}	DELETE		Experimenter UI	None

Analysis tool

- Lists all experiments
- For each experiment the analysis tool
    - fetches the experiment data from the server.
    - displays the experiment results as JSON.
    - gives four alternative forms for the JSON data.
        - The original experimental data is quite nested: contains almost everything about the experiment.
        - JSON data is simplified.
            - E.g. JSON is an array of key-value-experiment group ID objects.
    - creates a new JSON file in chosen form.