TWIQA

A DSL for developing Twitter Q&A Bots

Benedikt Langer, Lukas Schäfer, David Kleindiek

Table of Contents

Twitter Q&A games

- User follows bot
- Bot sends challenges
- User provides answers
- Bot checks answers

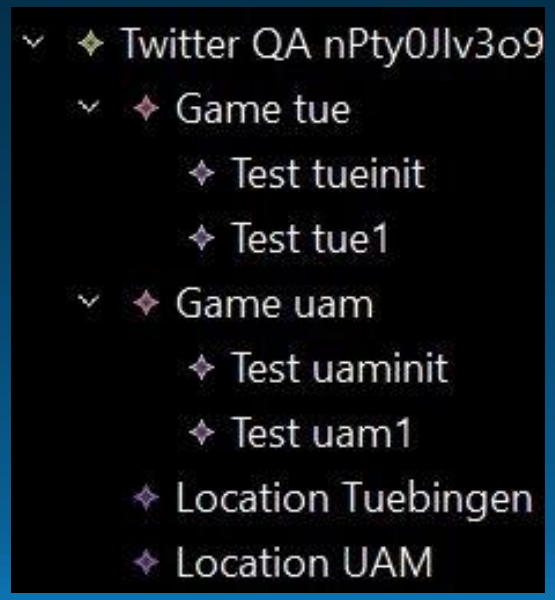
- Given locations
- Time limits / Attempts
- Set of answers
- Set of hints
- Reward → competitive

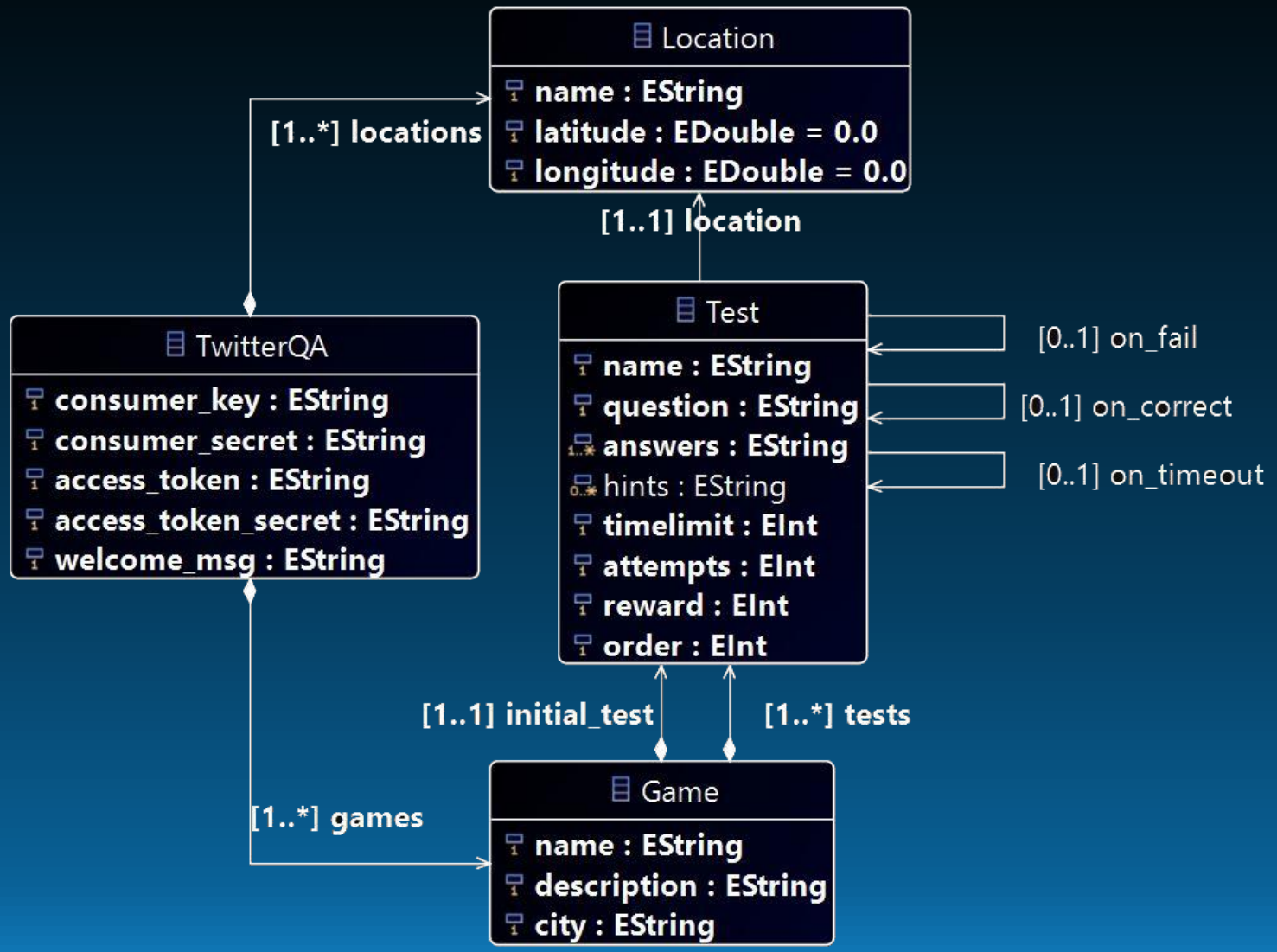
Developing stages

- Design a Metamodel
- Develop a textual DSL
- Implement the actual game → Twitter API
- Check games using petrinets

Metamodel

- Bot parameters
- Defines gamestructure





DSL: .twiqa

- X-Text
- Natural design
- Intuitive
- Brackets for structure
- Convert to .xmi for later use

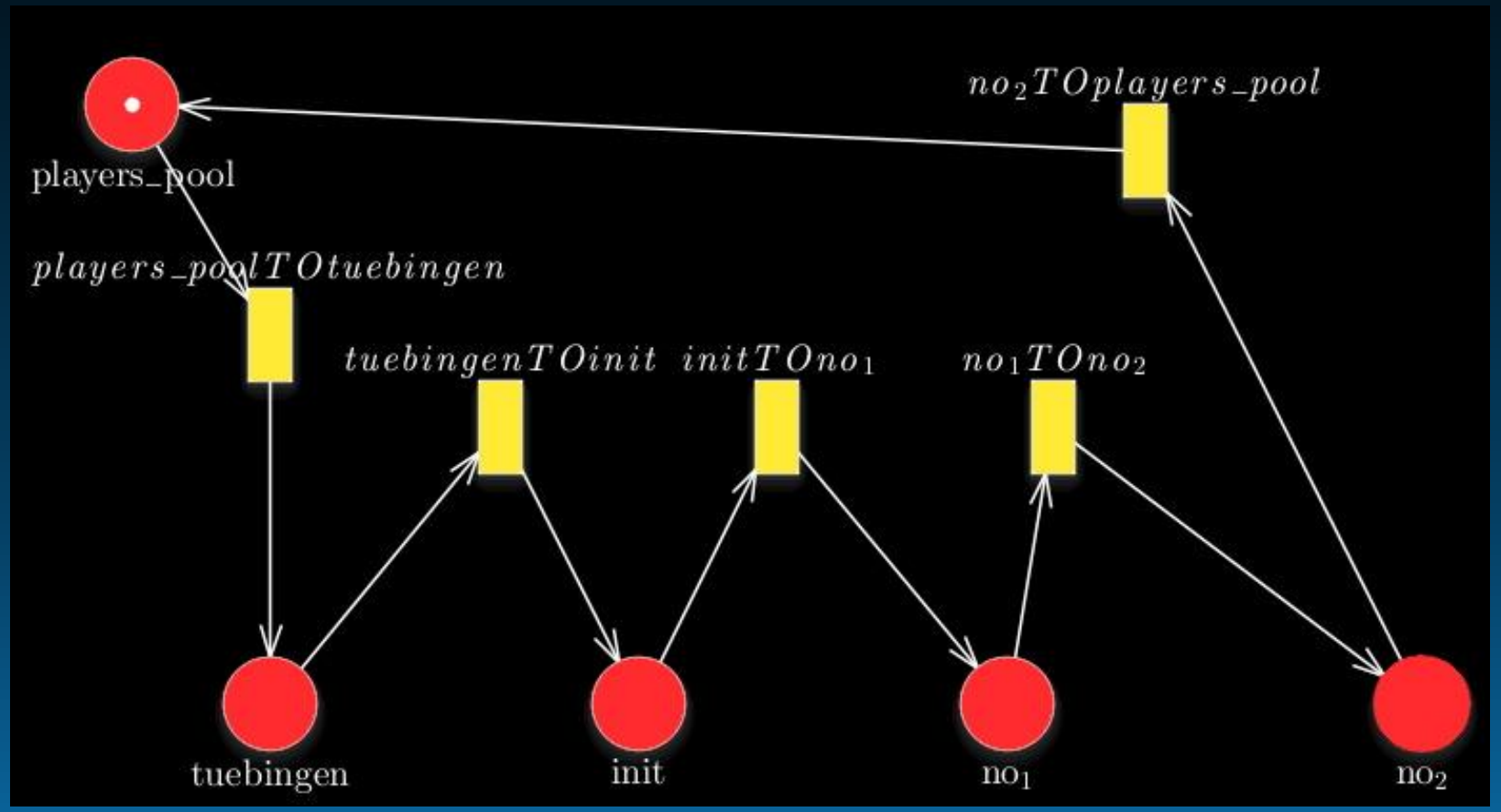
```
TwitterQA {
    consumer_key is 'key'
    consumer_secret is 'secret'
    access_token is 'token'
    access_token_secret is 'token_secret'
    welcome_msg is "WELCOME MSG"
    games are [
        Game NAME {
            description is "DESCRIPTION"
            city is "CITY"
            initial_test is Test NAME {
                question is "QUESTION"
                order is 0
                answers are ["", ""]
                hints are ["", ""]
                timelimit is 0
                attempts is 0
                reward is 0
                location is LOCATION
                on_correct is TEST_NAME
                on_fail is TEST_NAME
                on_timeout is TEST_NAME
            }
            tests are []
        }
    ]
    locations are [
        Location LOCATION {
            latitude is 283.2
            longitude is 1230.239
        }
    ]
}
```

Petrinets

- Game traversability
- Average game time

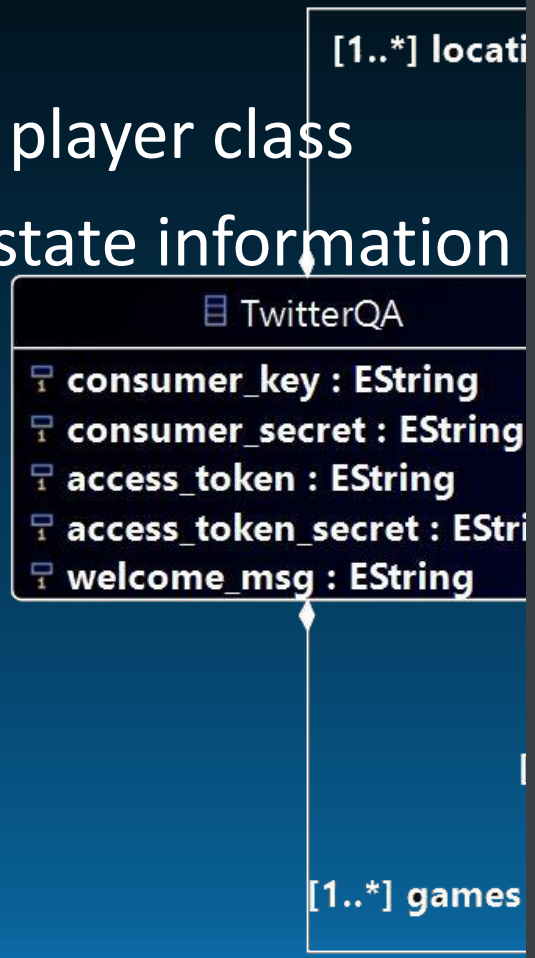
- Custom parser
- Convert xml structure

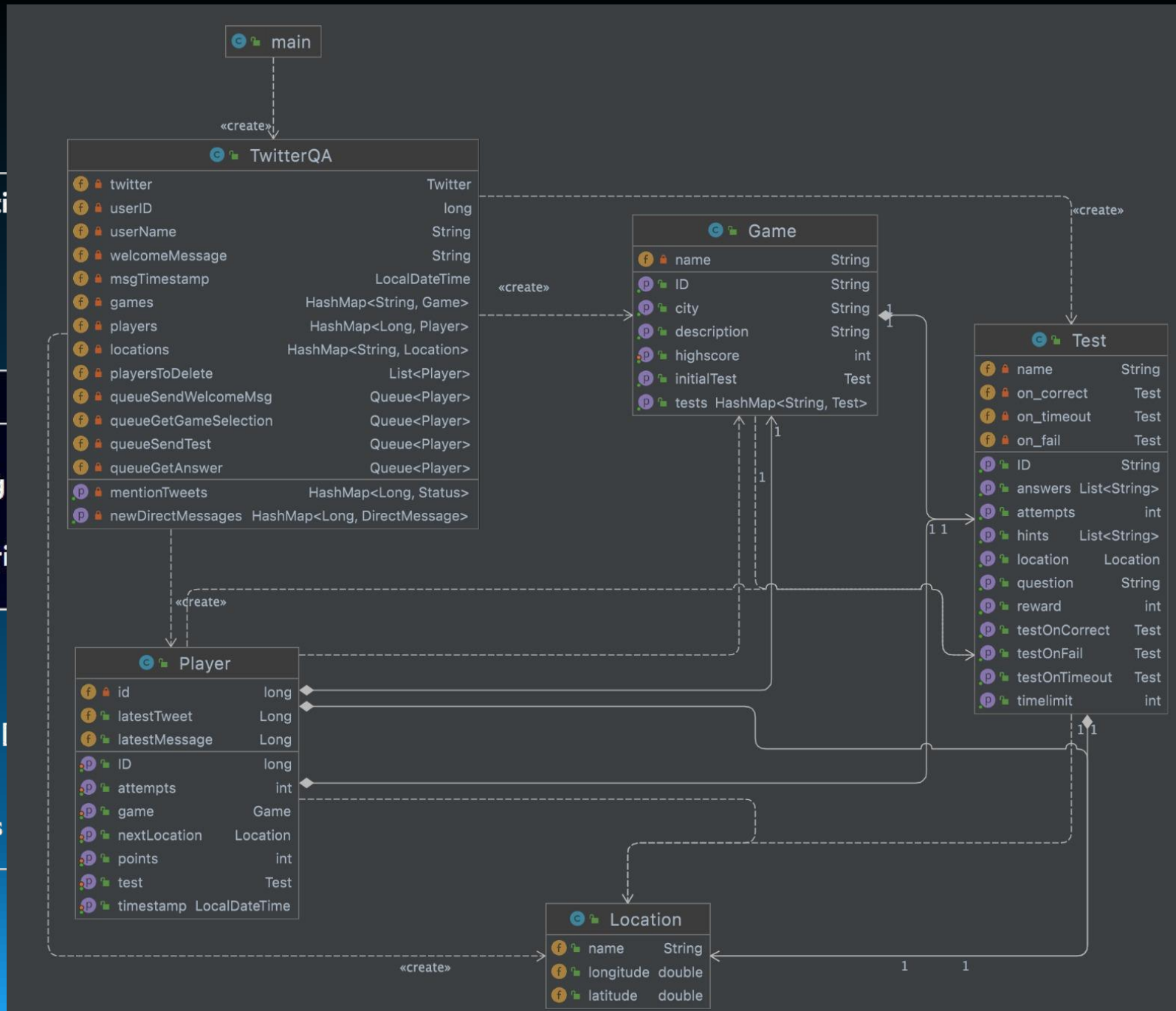
- GreatSPN



Java classes

- Added player class
- Game state information





Implementation

- Java with Twitter4j
- Controls account
- Event handler using queues
- Dynamic function ‘createGames’

Game loop

- SendWelcomeMsg
- GetGameSelection
- SendTest
- GetAnswer

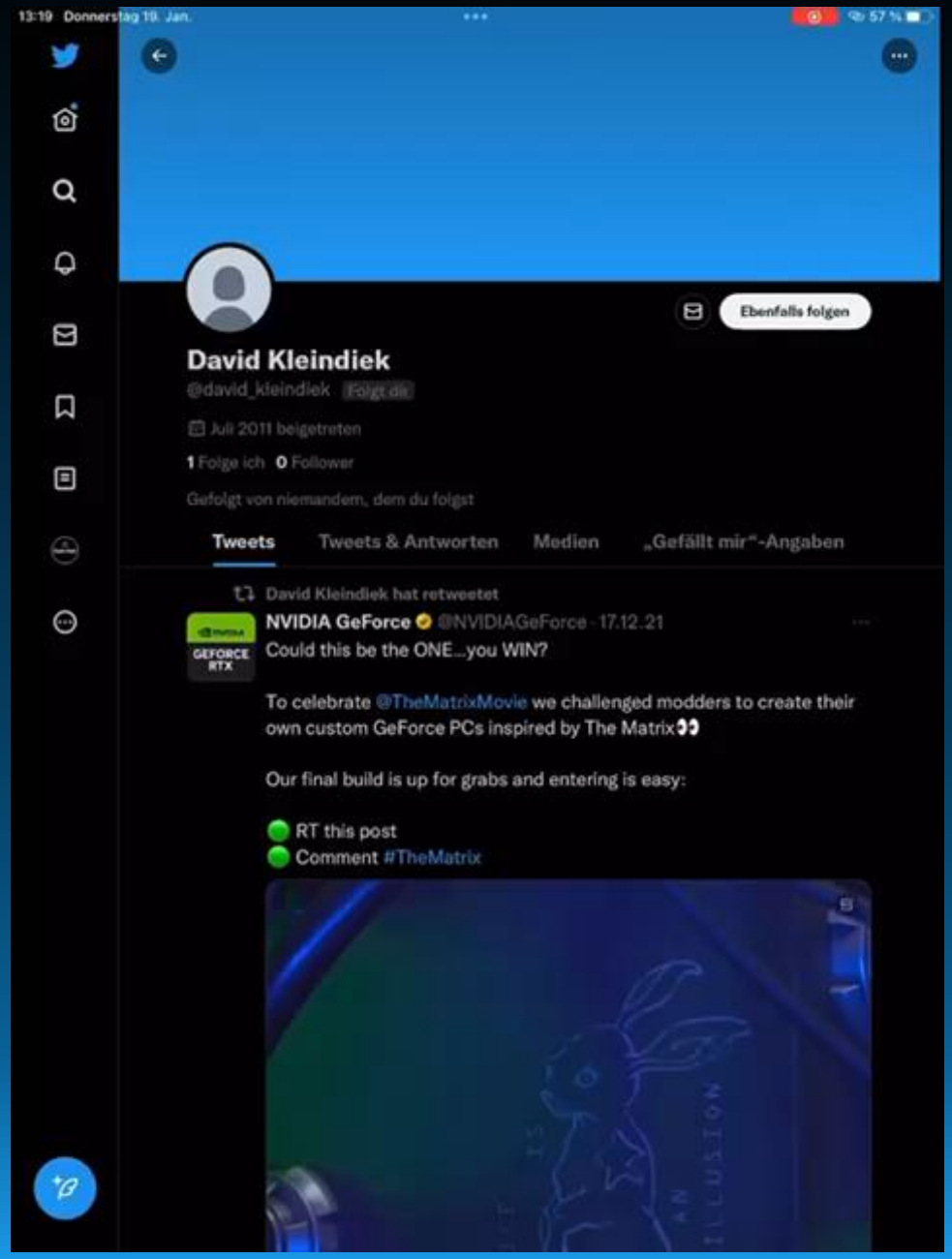
```
public void createGames() {
    StringBuilder sb = new StringBuilder();
    sb.append(welcomeMessage);

    [for (aLoc: Location | aTwitterQA.locations)]
    locations.put("[aLoc.name/]", new Location("[aLoc.n
    [/for]

    HashMap<String, Test> tests = new HashMap<String, T

    [for (aGame: Game | aTwitterQA.games)]
    [for (aTest: Test | aGame.tests->sortedBy(order)->r
    tests.put("[aTest.name/]", new Test("[aTest.name/]"
    [/for]
    Test [aGame.initial_test.name/] = new Test("[aGame.
    Game [aGame.name/] = new Game("[aGame.name/]", [aGa
    games.put([aGame.name/].getID(), [aGame.name/]);
    sb.append([aGame.name/].getID()).append(" - ").appe
    tests.clear();
    [/for]
    welcomeMessage = sb.toString();
}
```

Gameflow


13:19 Donnerstag 19. Jan.
Ebenfalls folgen
David Kleindiek
@david_kleindiek
Juli 2011 beigetreten
1 Folge ich 0 Follower
Gefolgt von niemandem, dem du folgst
Tweets
Tweets & Antworten
Medien
„Gefällt mir"-Angaben
David Kleindiek hat retweetet
NVIDIA GeForce @NVIDIAGeForce · 17.12.21
Could this be the ONE...you WIN?
To celebrate @TheMatrixMovie we challenged modders to create their own custom GeForce PCs inspired by The Matrix👀
Our final build is up for grabs and entering is easy:
RT this post
Comment #TheMatrix

Limitations and improvements

- Updates every minute → API limitation
- Friendship required for private accounts
- ‘;’ required in answer
- Location not accurate

- More bot commands
- Require image at location
- Calculate avg gametime

Challenges

- Collaboration problems with Eclipse
- Restructuring of model is complex
- Insufficient API documentation

But we did it!

Questions?

https://github.com/stedavkle/twitter-q-a