Studiengangsabend: Informatik

Live-Coding @ HSR

Martin Kempf, Reto Kleeb

September 2015

ELCA We make it work.

1. Who are we?
2. Sudoku Battle Royal: The Game Idea
3. Technology-Stack
4. Live-Coding
5. Deployment
6. Live-Gaming

WHO ARE WE?

Martin Kempf

Senior Software Engineer

At ELCA since 2013

BSc in Computer Science: Hochschule f. Technik Rapperswil 2008

MSc in Computer Science: Hochschule f. Technik Rapperswil 2012

Reto Kleeb

Senior Software Engineer

At ELCA since 2012

BSc in Computer Science: Hochschule f. Technik Rapperswil 2008

MSc in Computer Science: Northeastern University 2012

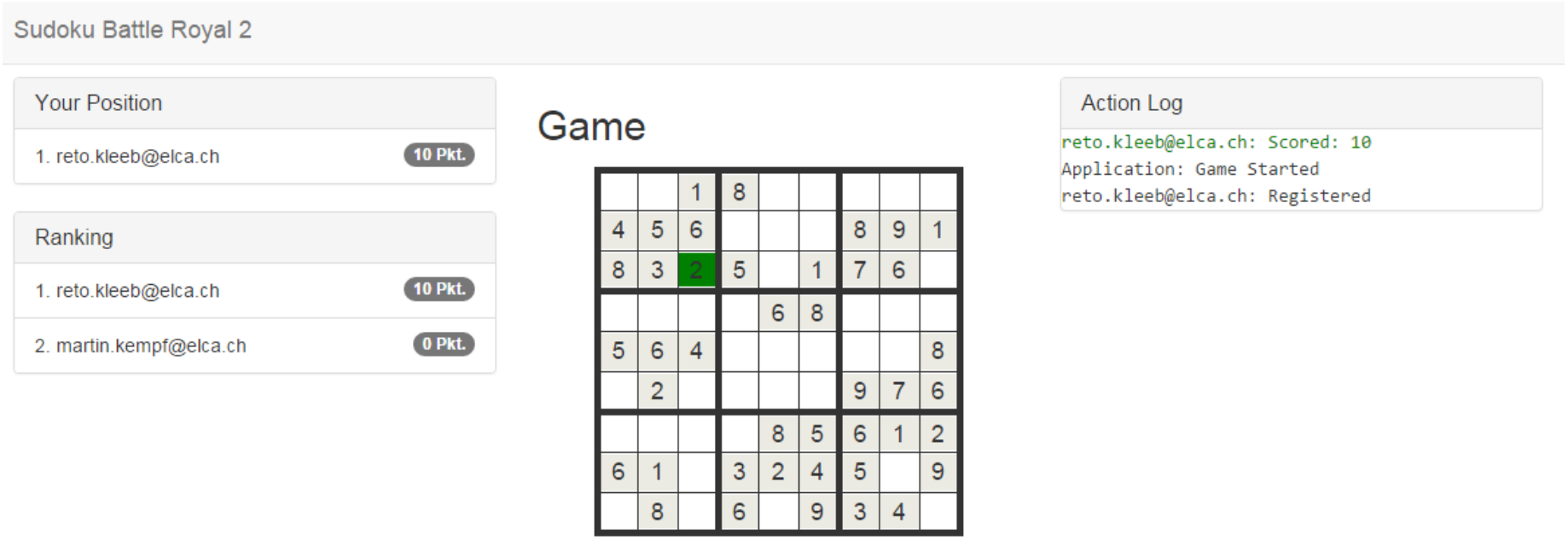
GAME IDEA

- Competitive / Concurrent Solving of a Shared Sudoku Puzzle:
  - If a participant enters a correct value, this value is shared with all other participants (field becomes read-only)
  - Correct Values are rewarded with +6 Points
  - Invalid Values are punished with -2 Points
- The game is over as soon as there are no empty fields left.

- Based on SE2-Project
- Sudoku Battle Royal (V1)
  - Swing
  - Manual Socket-Handling / Protocol
  - Year ~2007…

GAME IDEA

- Sudoku Battle Royal (V2):
  - Browser Based Client (JS)
  - Communication trough Websockets
  - Simple Webserver based on Spring Boot



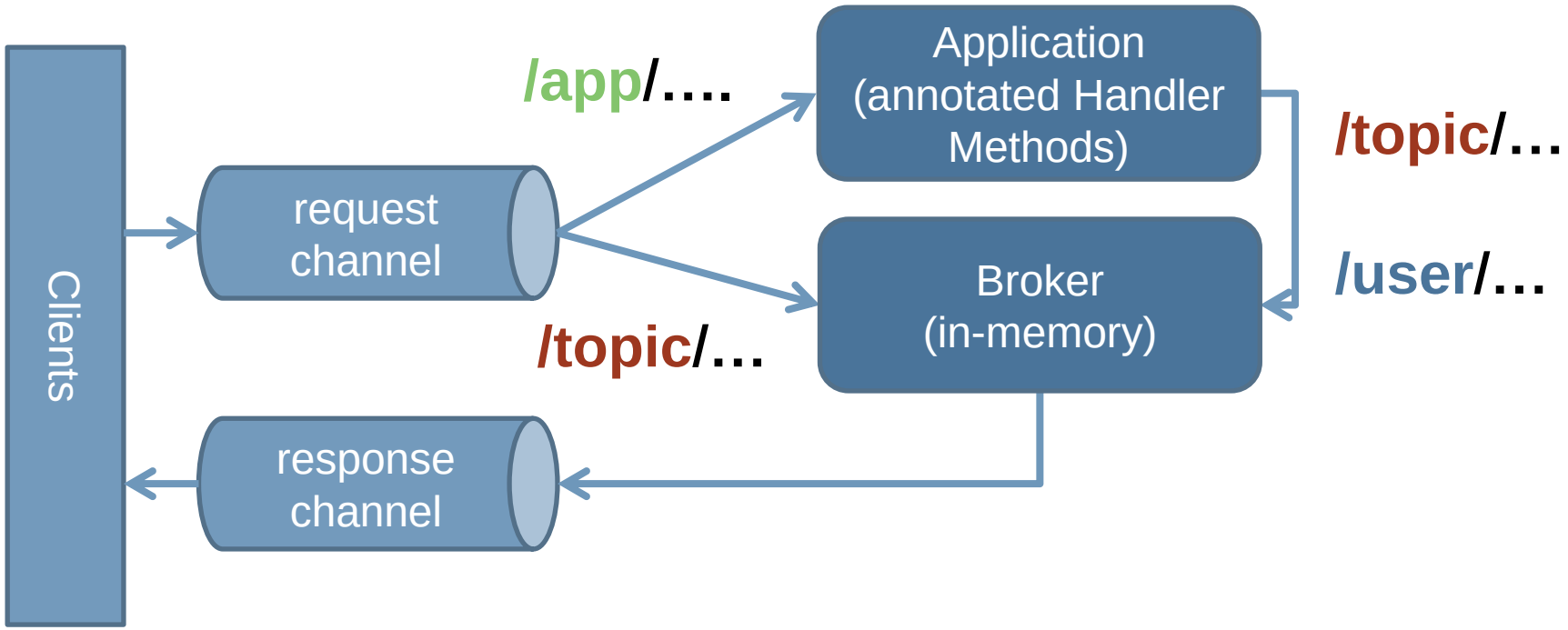
SOLUTION: MESSAGING OVER WEBSOCKETS

- Websockets
  - Standing Socket between Browser und Webserver
  - Duplex Communication
  - Supported in all modern Browsers and Servers
  - Allows simple server to client notifications (as opposed to classic HTTP)

- Our Solution (Messaging)
  - Use a message-based protocol on top of the websocket
  - Messaging conventions/protocol simplify client/server communication
  - Broker provides support for topic registrations and redistribution

MESSAGING: BUILDING BLOCKS

- Subscribe to a topic
  - Public topics
  - Personal topics
- Send Messages to a destination
- On top of Websocket “/socksjsendpoint”

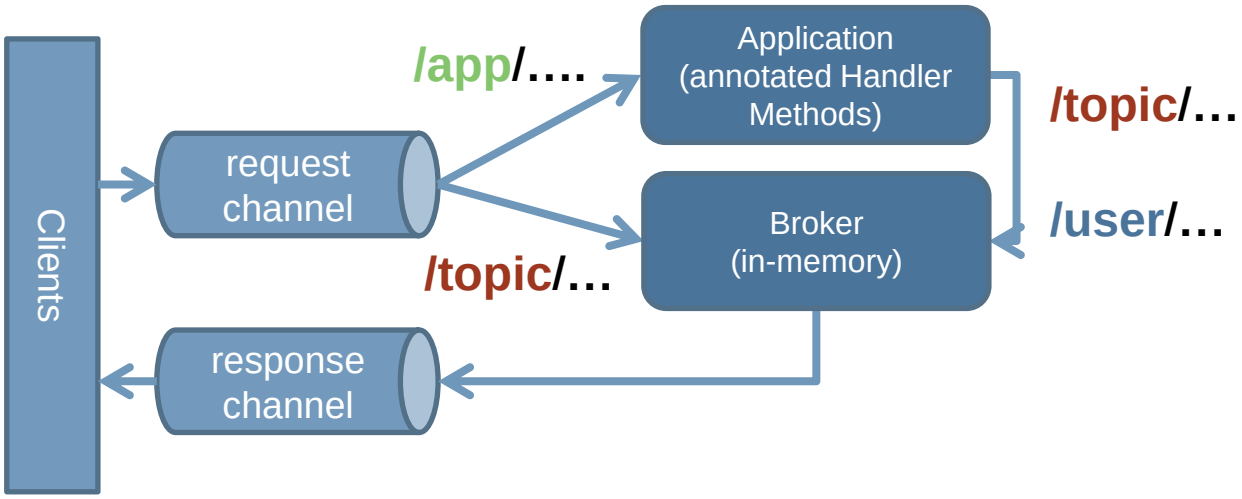


MESSAGING: BUILDING BLOCKS

```
>>> SUBSCRIBE
id:sub-0
destination:/topic/game/login
```

```
<<< MESSAGE
destination:/topic/game/login
content-type:application/json;charset=UTF-8
subscription:sub-0
message-id:n_tlbiyq-1
content-length:37

{"playerName":"martin.kempf@elca.ch"}
```



TECHNOLOGY: JAVASCRIPT / WEBSOCKET SUPPORT

sock.js

- SockJS provides a “WebSocket-like object”
- Consistent Socket API in all Browsers and network environments
- https://github.com/sockjs/sockjs-client

stomp.js

- STOMP: Simple (or Streaming) Text Orientated Messaging Protocol
- stomp.js allows us to easily use STOMP on top of a Websocket
- http://jmesnil.net/stomp-websocket/doc/

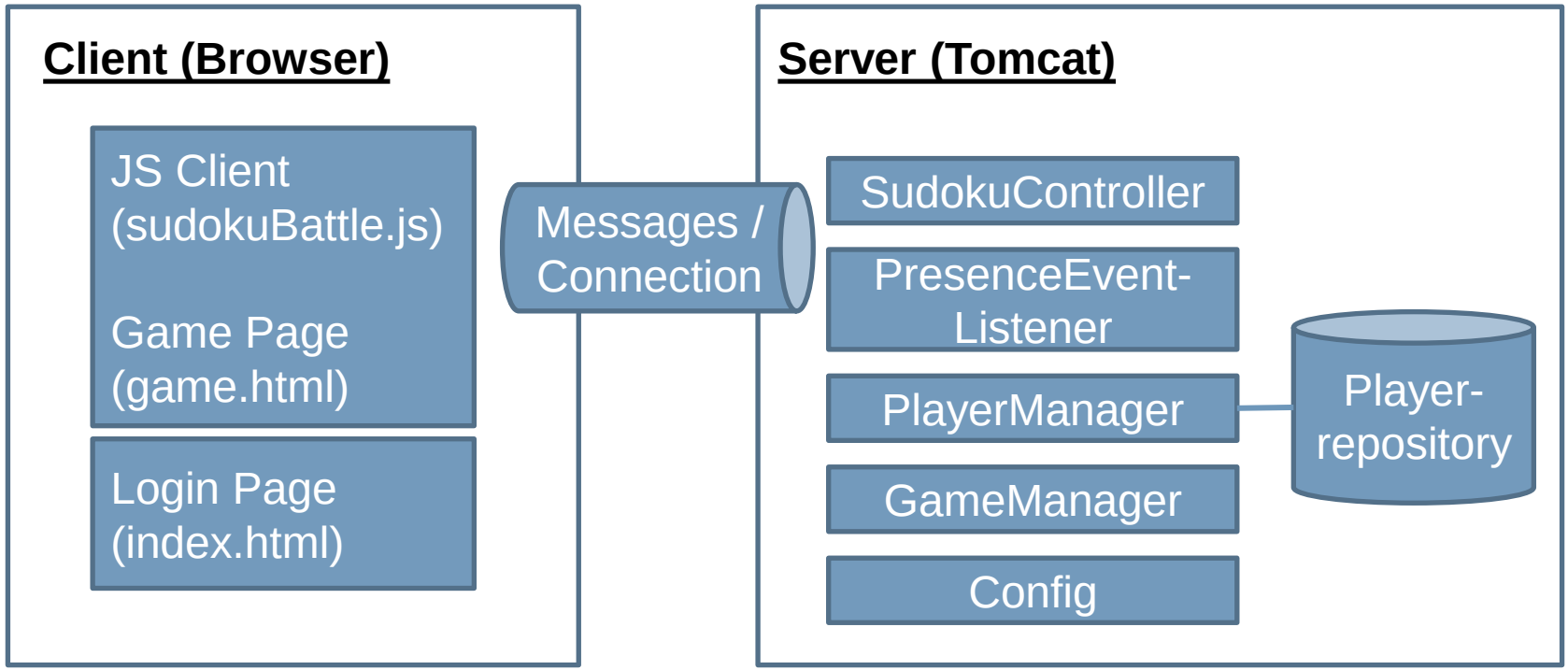
vue.js

- Links View and Model via two way data bindings (similar to the AngularJS bindings)
- DOM manipulations are abstracted away
- http://vuejs.org

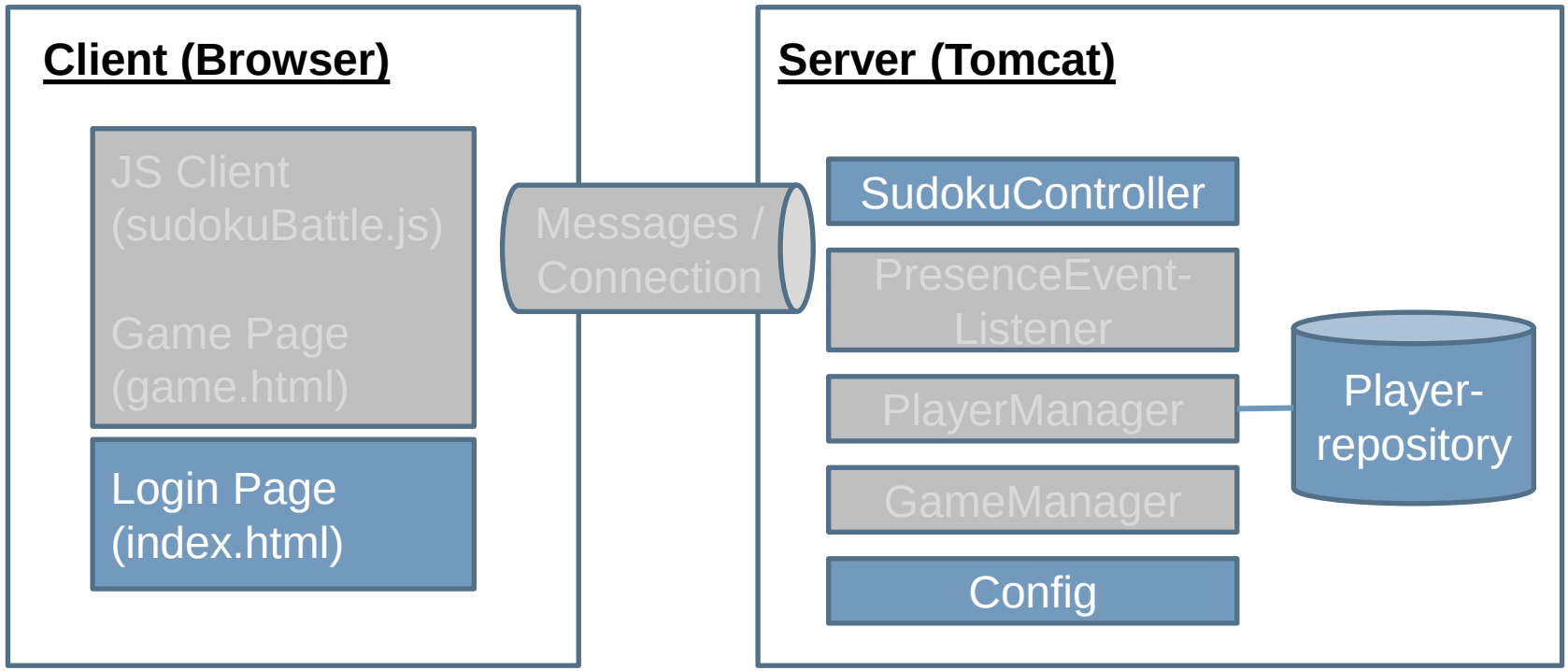
TECHNOLOGY: SPRING BOOT

- Spring (Boot)
  - Dependency Injection
  - Simple REST Endpoints
  - Spring Security for authentication
  - Simple In-Memory Persistence out of the Box (Spring Data JPA)
  - In-Memory Broker for STOMP Messages

APPLICATION OVERVIEW

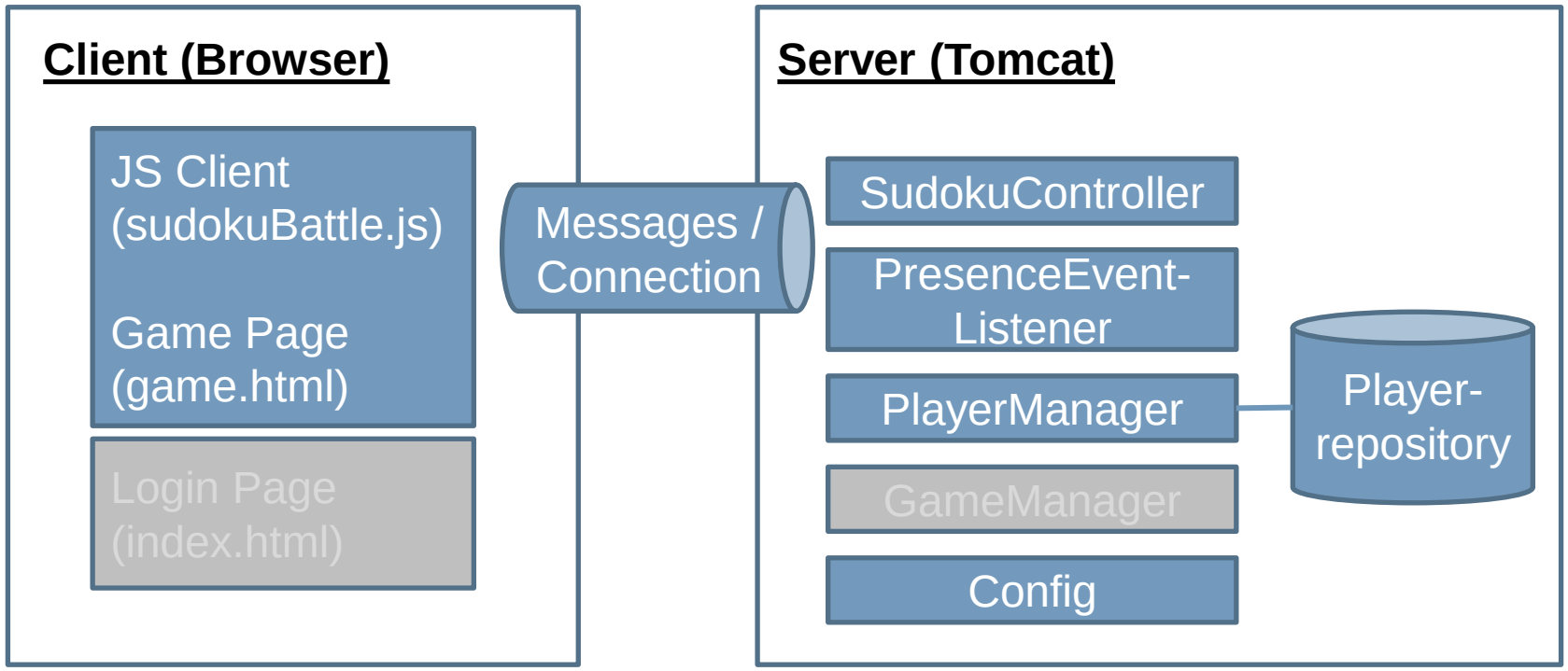


STAGE 0: BASIC SETUP



- Plain Spring Boot Scaffold that displays static HTML page
- Basic Spring Security Config
- Empty Controller for incoming messages
- Repository to persist Players

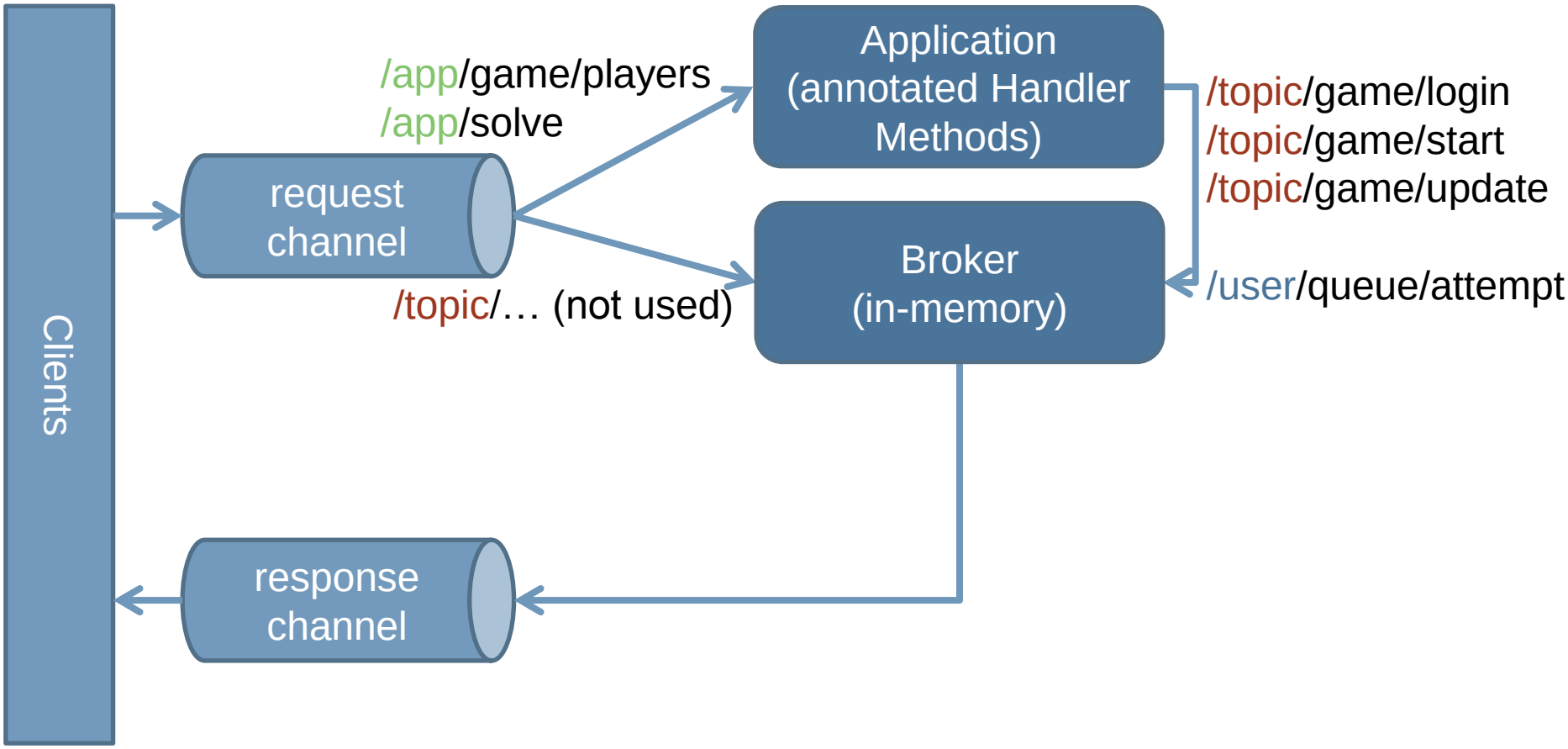
STAGE 1: SOCKET SETUP



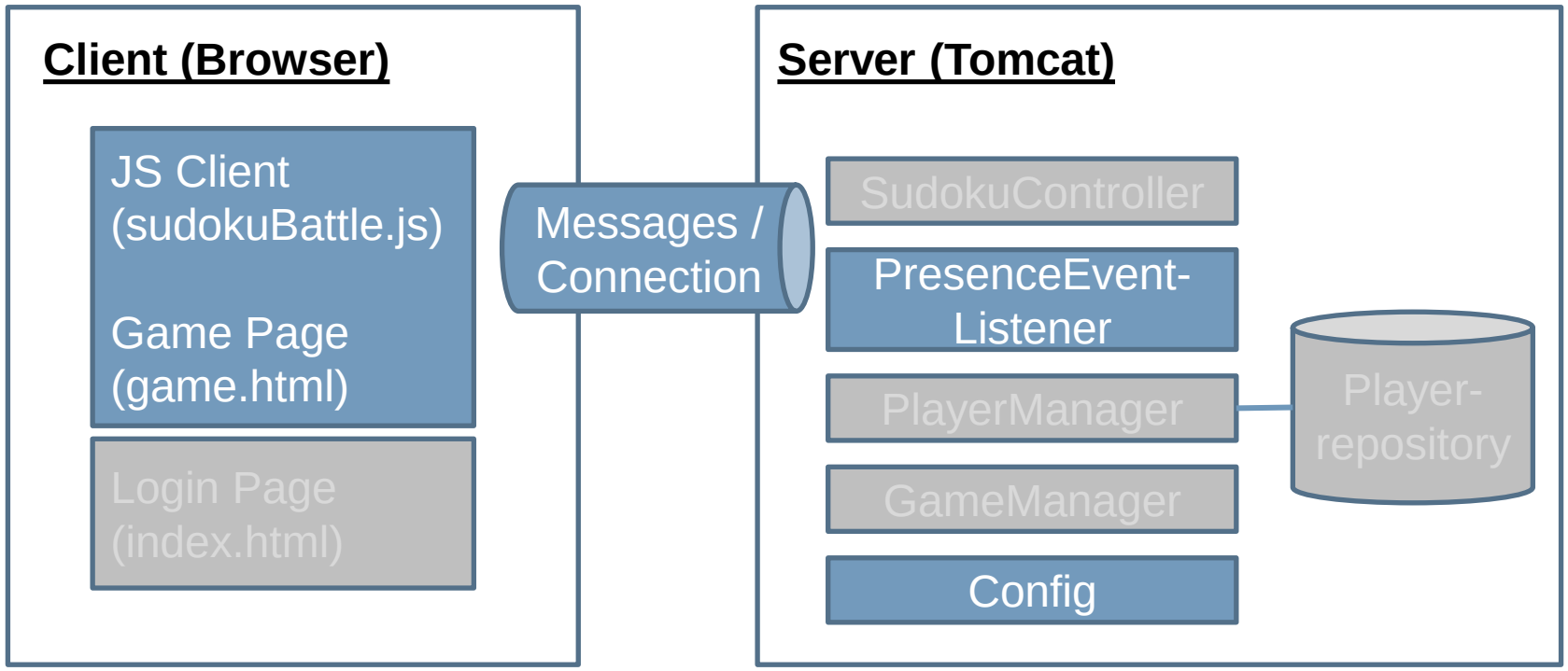
- Setup of Socket-Endpoint and Message-Broker
- Handling of “Session Connect Event” (stores Players in the Repo)
- Return a list of all (already) connected players on connect
- Client: Connect to Socket and subscribe to destination

MESSAGING: BUILDING BLOCKS

- On top of Websocket “/socksjsendpoint”

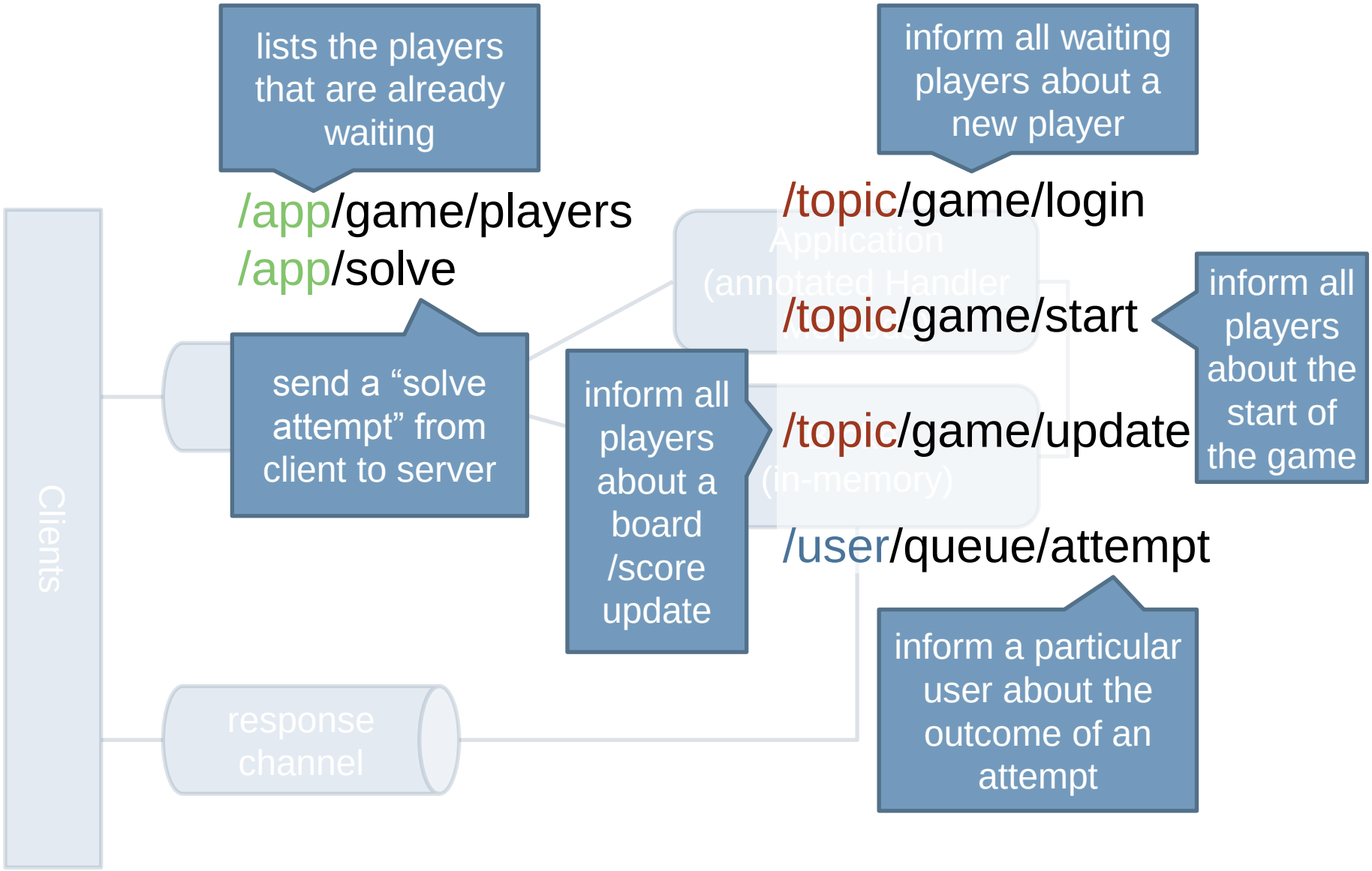


STAGE 2: SENDING MESSAGES FROM THE SERVER

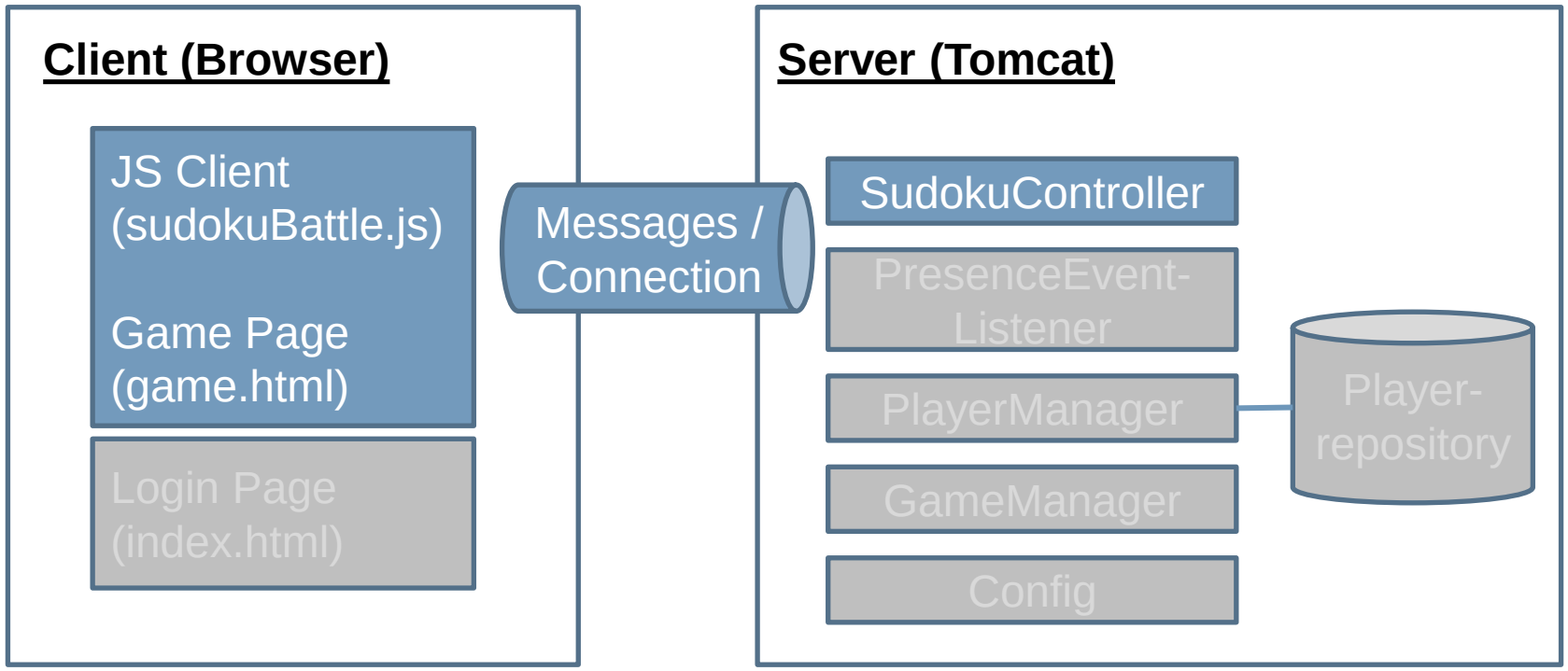


- Setup of Broker for “/topic” prefixed destinations
- Sending of messages to “/topic/game/login” after new session connections
- Client: Subscription to destination “/topic/game/login”

MESSAGING: BUILDING BLOCKS



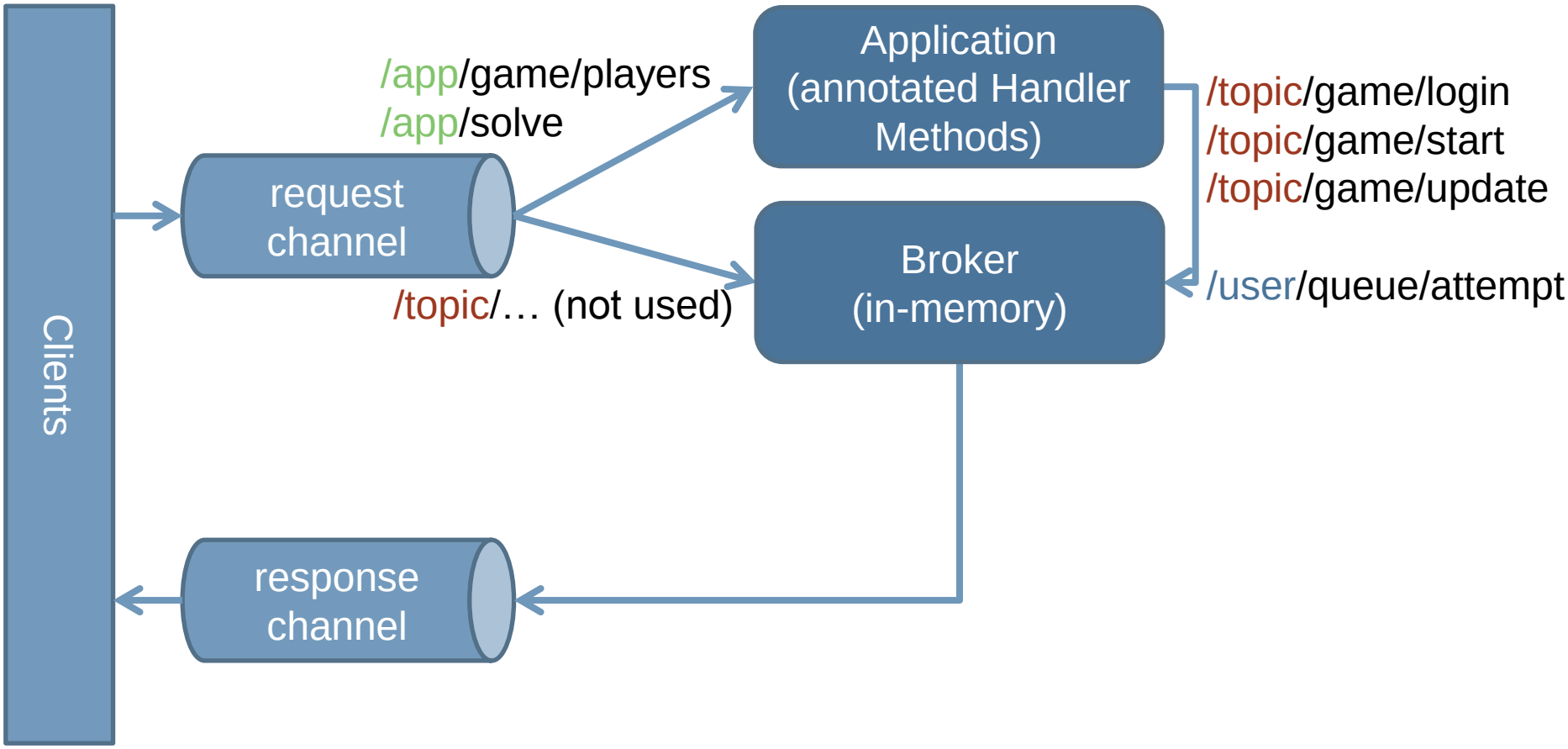
STAGE 3: SENDING MESSAGES FROM THE CLIENT



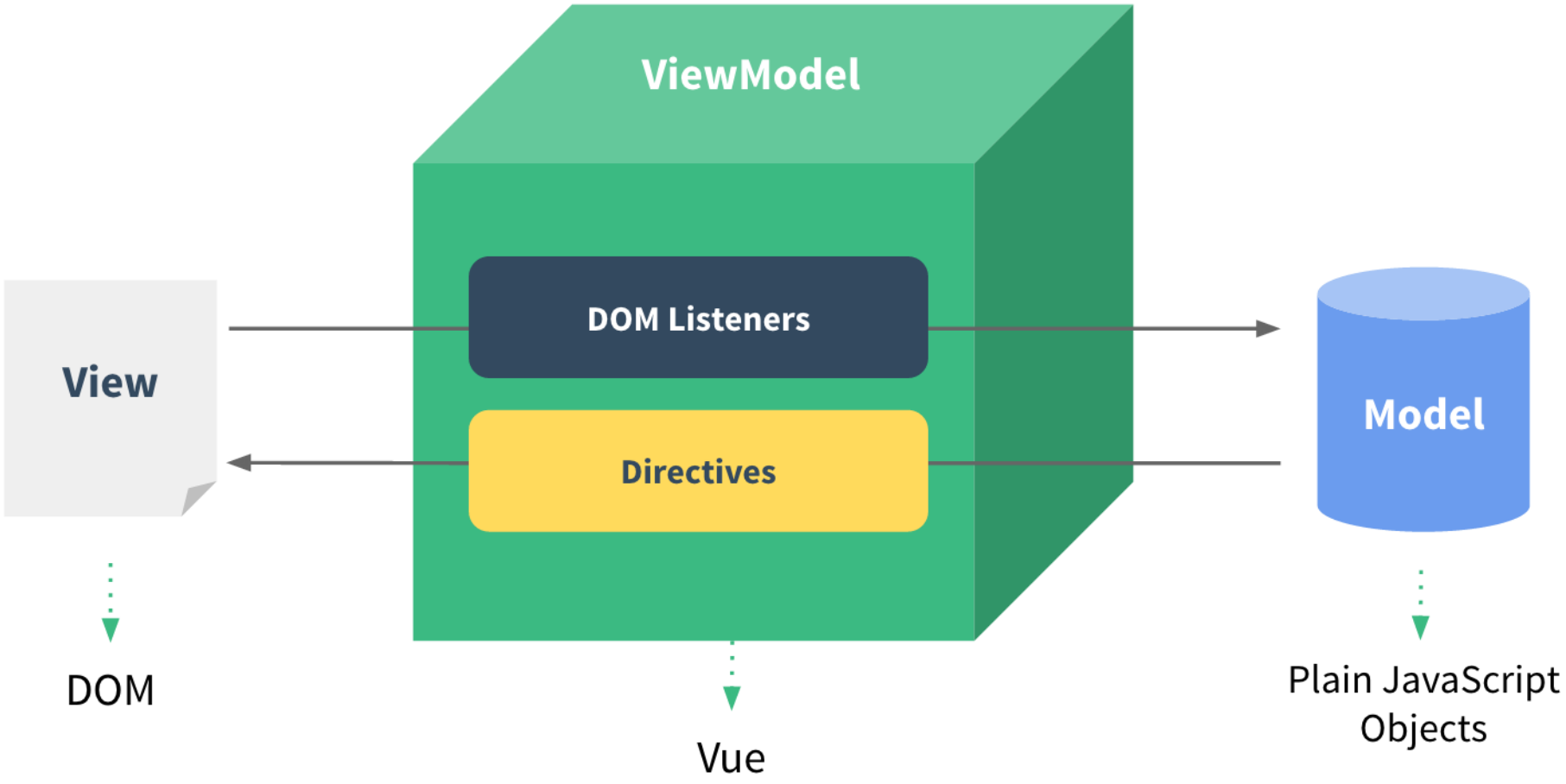
- Handling of “Solve” Messages on the Server
  - Messages to public destination
  - Messages to personal destination
- Sending “Solve Messages” from the Client

MESSAGING: BUILDING BLOCKS

- On top of Websocket “/socksjsendpoint”

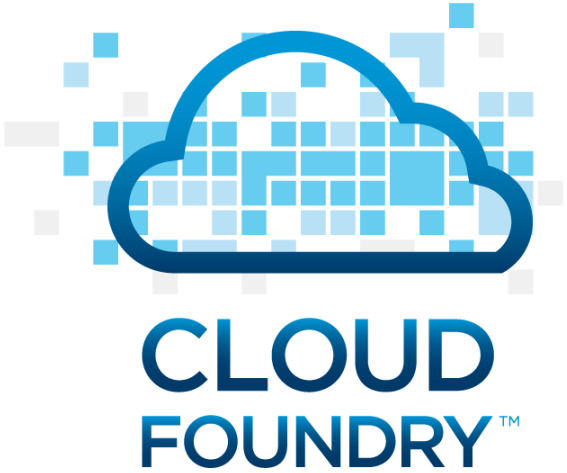


STAGE 4: BINDING THE MODEL TO THE VIEW ON THE CLIENT



DEPLOYMENT

- Compilation and Packaging (using Maven)
- Upload of the JAR file to a PAAS Provider
  (using the «cf» command line tool)



THE “OFFICIAL” RULES

- All participating players are divided into 3-5 player-groups for the 3-5 elimination rounds
- All participating players *(of the current elimination round)* log in (using their email address)
- The administrator starts the game
- Once a game has been started, no new players can connect
- Once the puzzle has been completely solved the game is over
- The top 2 players of each elimination round advance to the final round
- The player with the top score in the final round wins the price
- In case of a tie we will draw a winner

- **http://battletest.cfapps.io/**

URLS

- Github Repository: https://github.com/ret0/sudokuBattleRoyal2
  - Contains the complete sources and these slides

- Related Examples and Tutorials:
  - https://spring.io/guides/gs/messaging-stomp-websocket/
  - https://github.com/salmar/spring-websocket-chat
  - http://g00glen00b.be/spring-angular-sockjs/

Thank you.

Contact

Martin Kempf
Senior Software Engineer
martin.kempf@elca.ch

Reto Kleeb
Senior Software Engineer
reto.kleeb@elca.ch

ELCA Informatique SA | Lausanne 021 613 21 11 | Genève 022 307 15 11
ELCA Informatik AG | Zürich 044 456 32 11 | Bern 031 556 63 11

www.elca.ch

| © ELCA