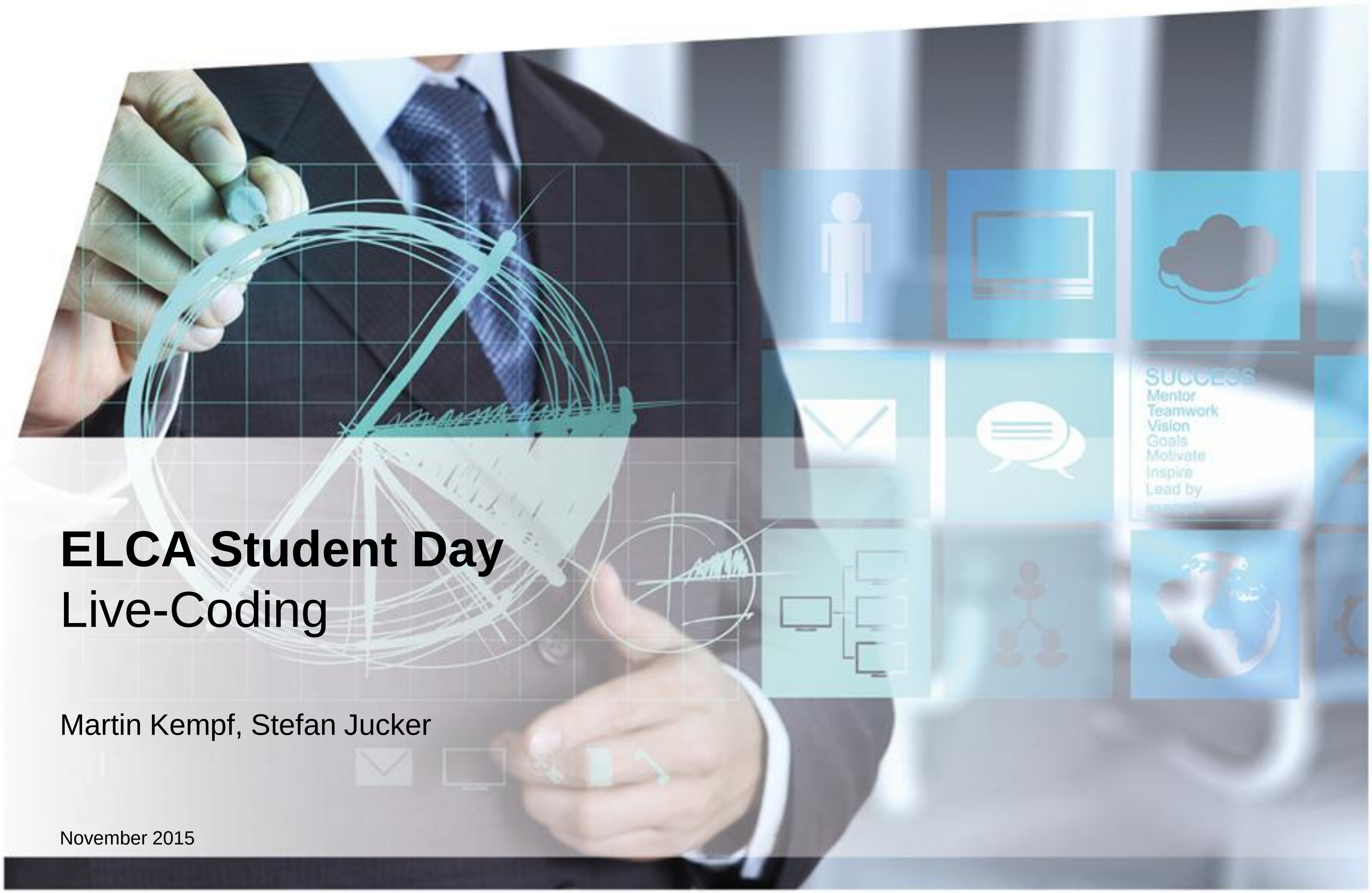


ELCA Student Day
Live-Coding

Martin Kempf, Stefan Jucker

November 2015

ELCA *We make it work.*

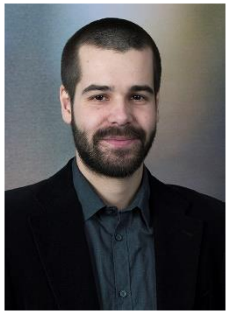
WHO ARE WE?

Martin Kempf

Software Engineer

At ELCA since 2013

MSc in Computer Science: Hochschule f. Technik Rapperswil 2012



Stefan Jucker

Software Engineer

At ELCA since 2013

MSc in Computer Science: ETH Zürich 2012

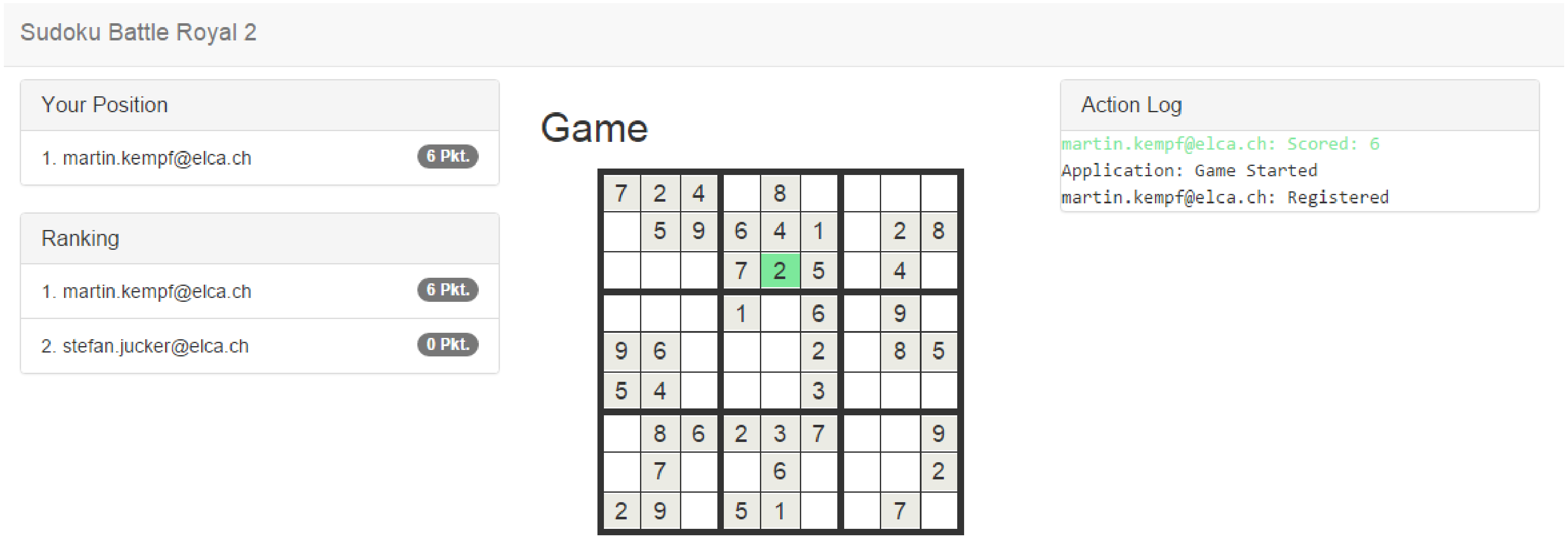
GAME IDEA

- 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

- Motivation
  - Originally for presentation at HSR
  - Similar to mini projects to evaluate new technologies (Java CC)

GAME IDEA

- Sudoku Battle Royal:
  - Browser-based client (JS)
  - Communication through WebSockets
  - Simple webserver based on Spring Boot



Sudoku Battle Royal 2

Your Position

1. martin.kempf@elca.ch 6 Pkt.

Ranking

1. martin.kempf@elca.ch 6 Pkt.
2. stefan.jucker@elca.ch 0 Pkt.

Game

7	2	4		8				
	5	9	6	4	1		2	8
			7	2	5		4	
			1		6		9	
9	6				2		8	5
5	4				3			
	8	6	2	3	7			9
	7			6				2
2	9		5	1			7	

Action Log

```
martin.kempf@elca.ch: Scored: 6
Application: Game Started
martin.kempf@elca.ch: Registered
```

ELCA

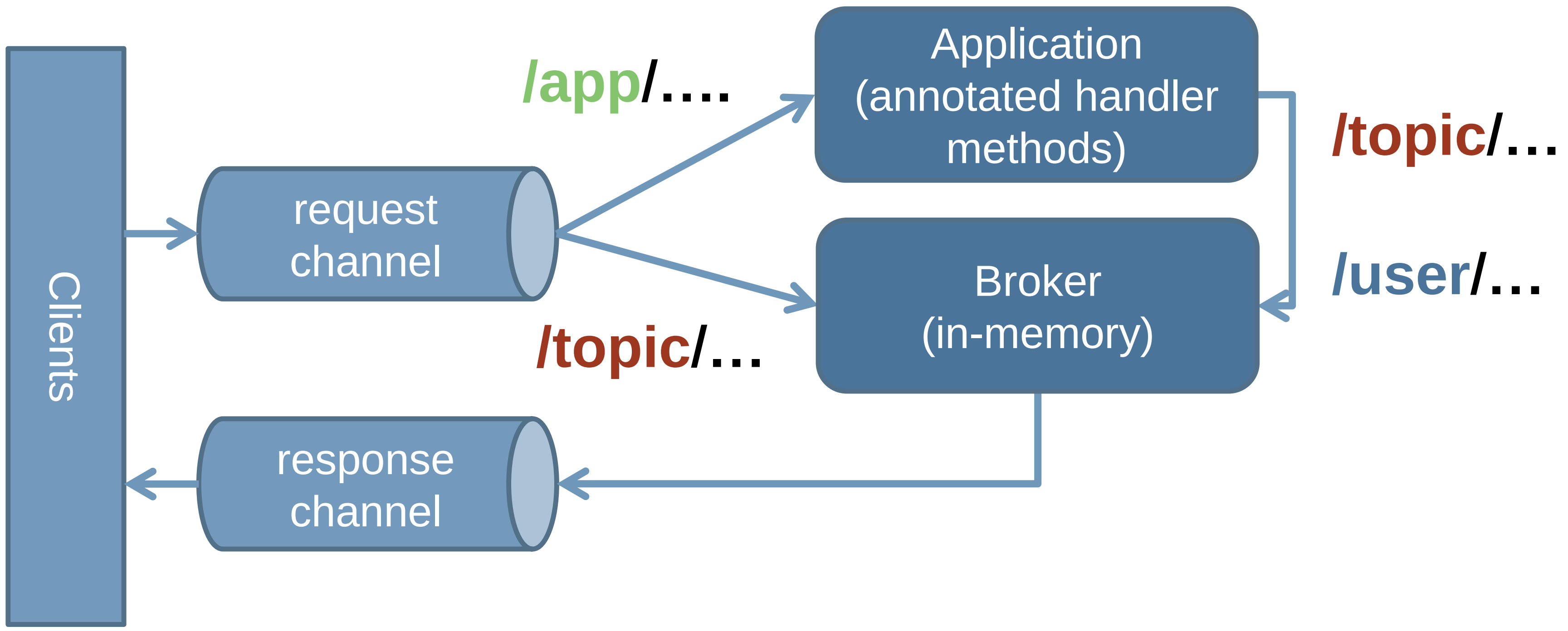
SOLUTION: MESSAGING OVER WEBSOCKETS

- 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)

- 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”

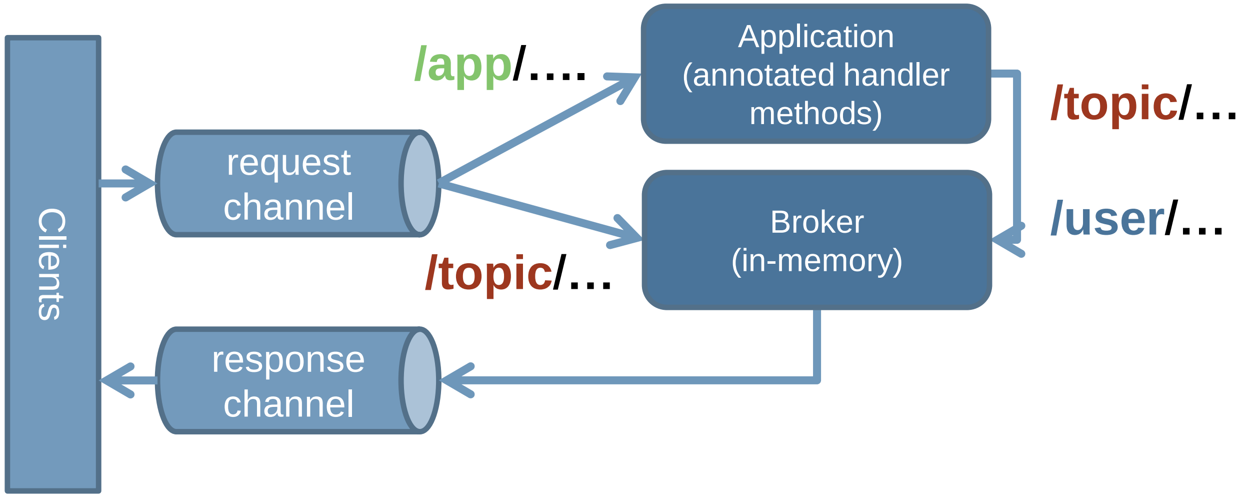


MESSAGING: BUILDING BLOCKS

```
>>> 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/

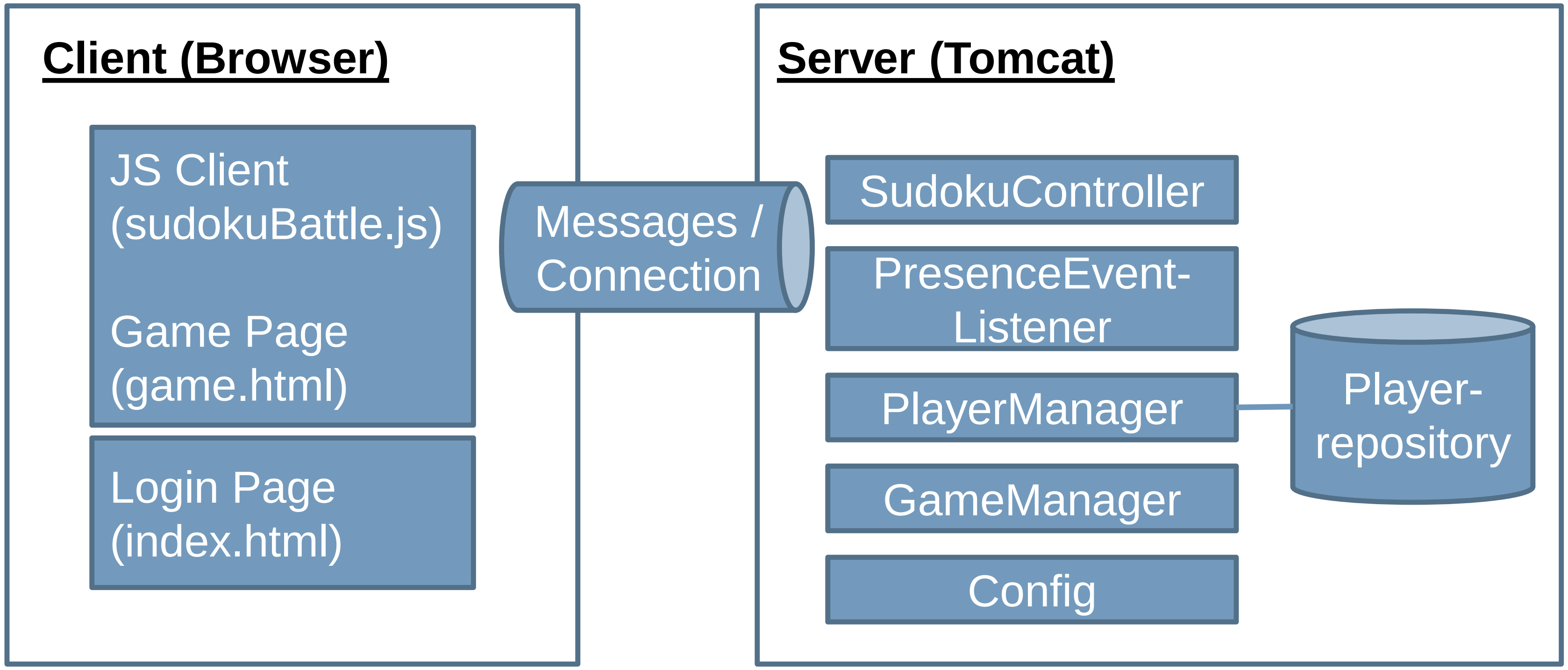
vue.js

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

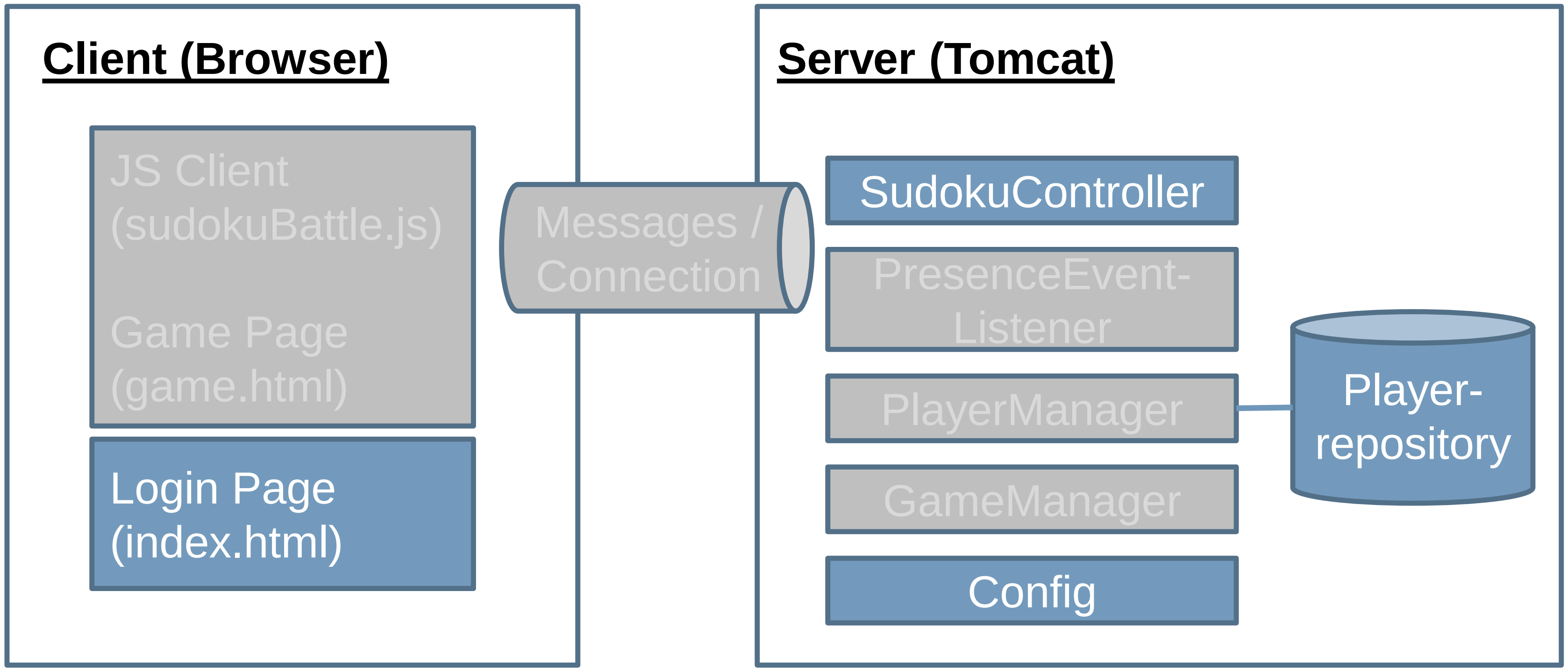
TECHNOLOGY: SPRING BOOT

- 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
  - Stand-alone Spring application

APPLICATION OVERVIEW

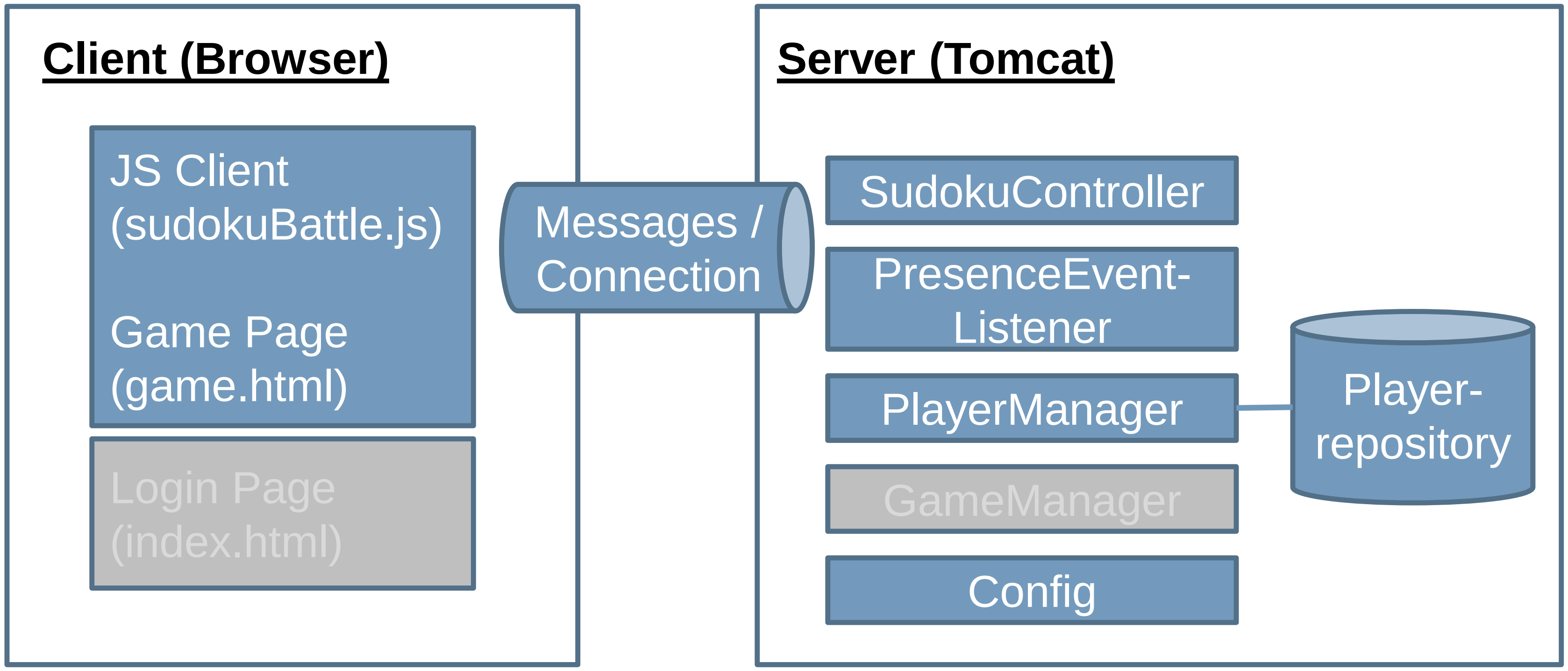


STAGE 0: BASIC SETUP



- Plain Spring Boot scaffold that displays static HTML page
- Basic Spring Security configuration
- Empty controller for incoming messages
- Repository to persist players

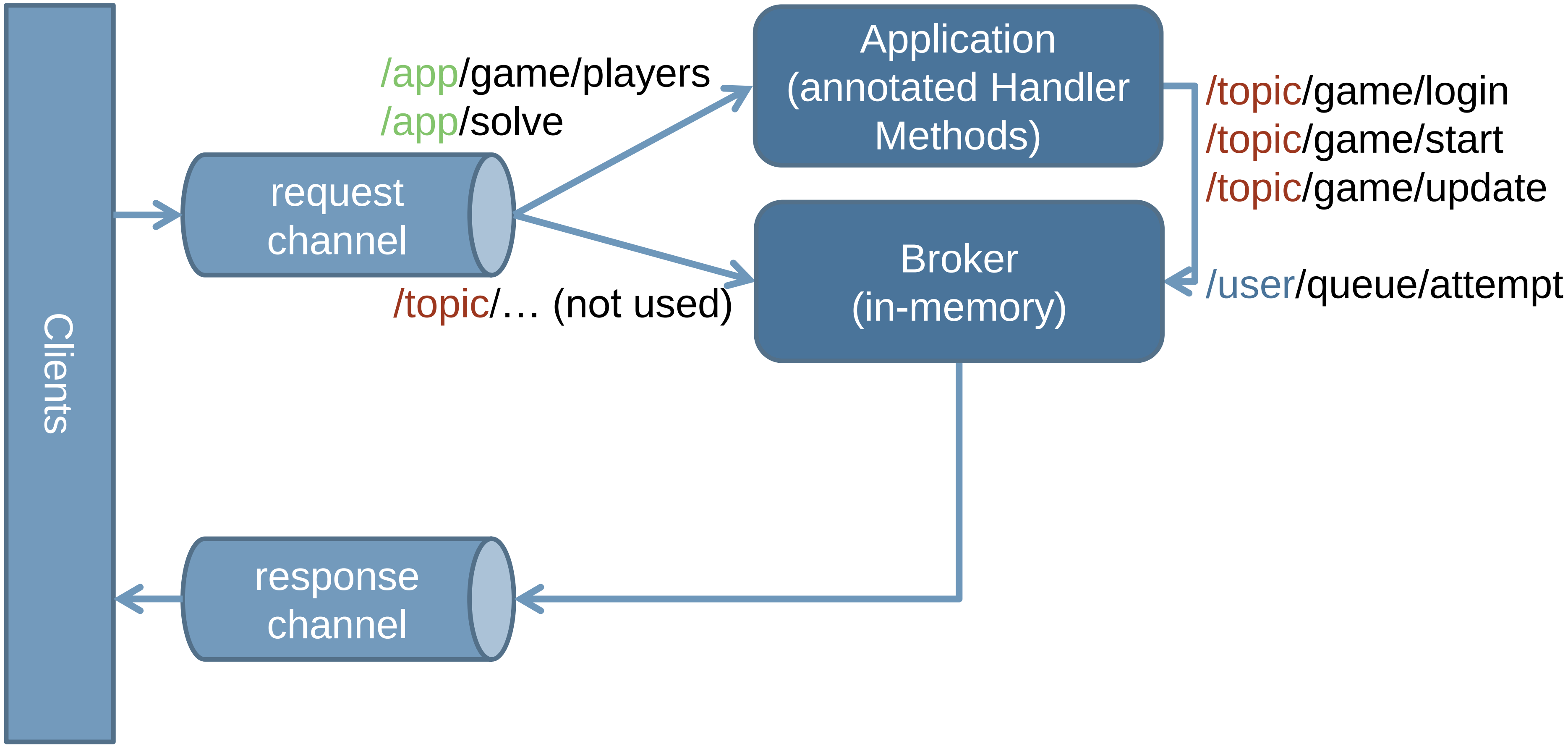
STAGE 1: SOCKET SETUP



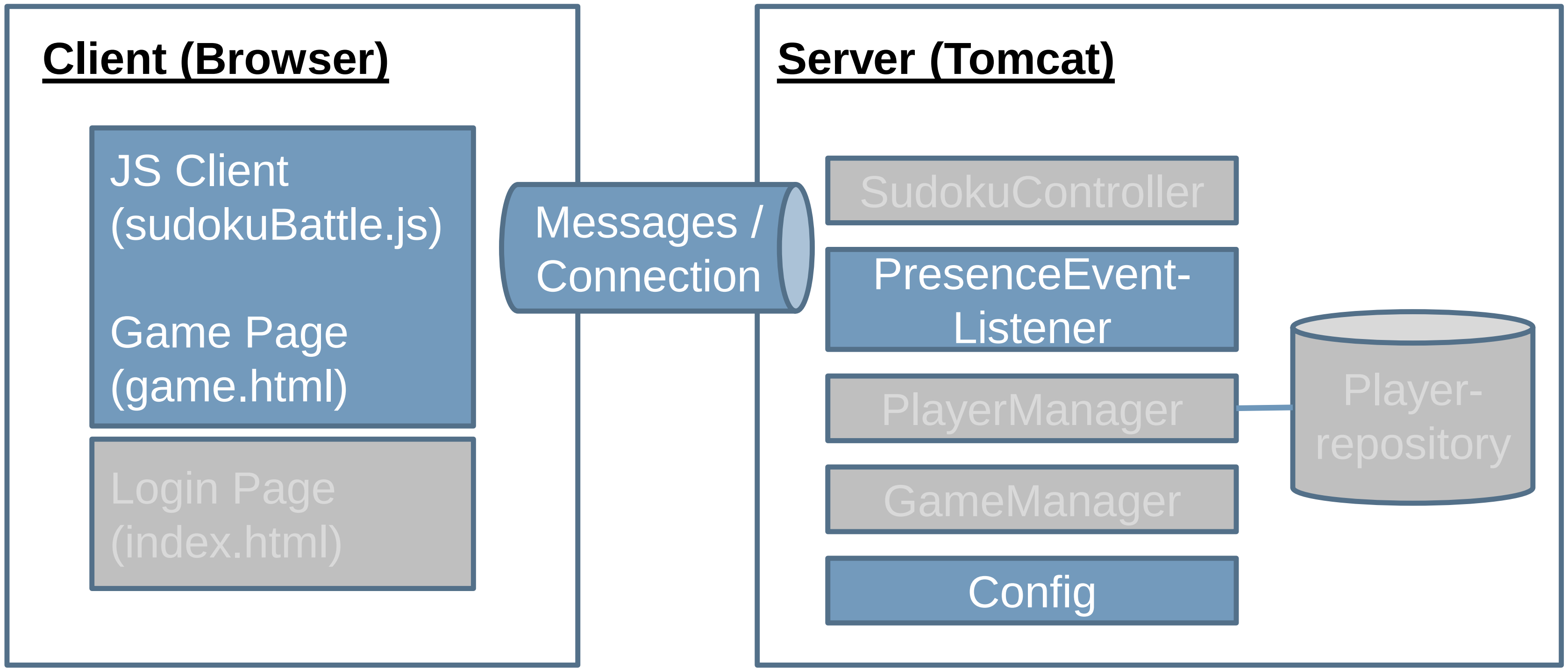
- Setup of socket-endpoint and message-broker
- Handling of “SessionConnectEvent” (stores players in the repo)
- Return a list of all (already) connected players on connect
- Client: connect to socket and subscribe to destination

MESSAGING: BUILDING BLOCKS

- On top of WebSocket “/socksjsendpoint”

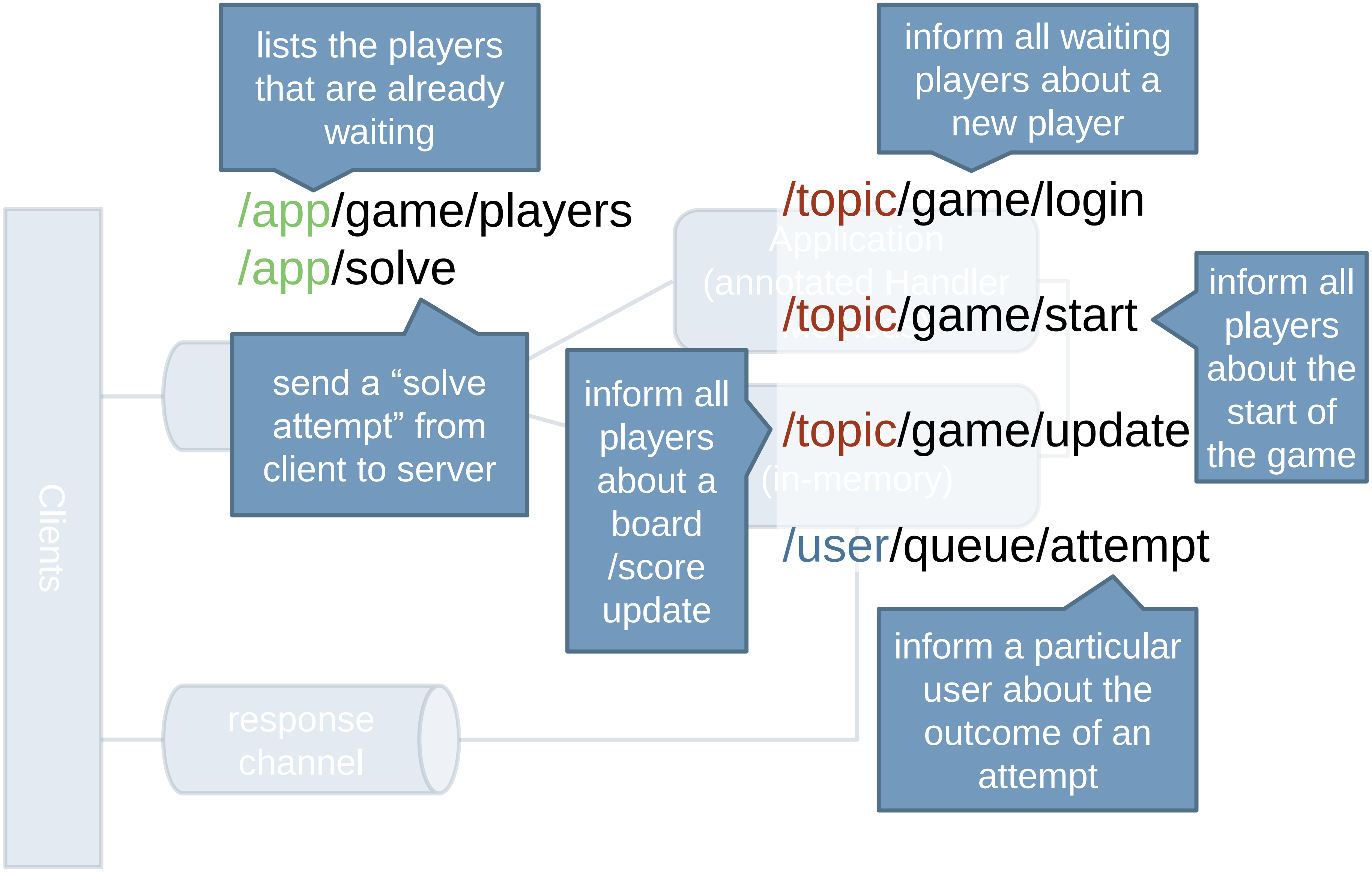


STAGE 2: SENDING MESSAGES FROM THE SERVER



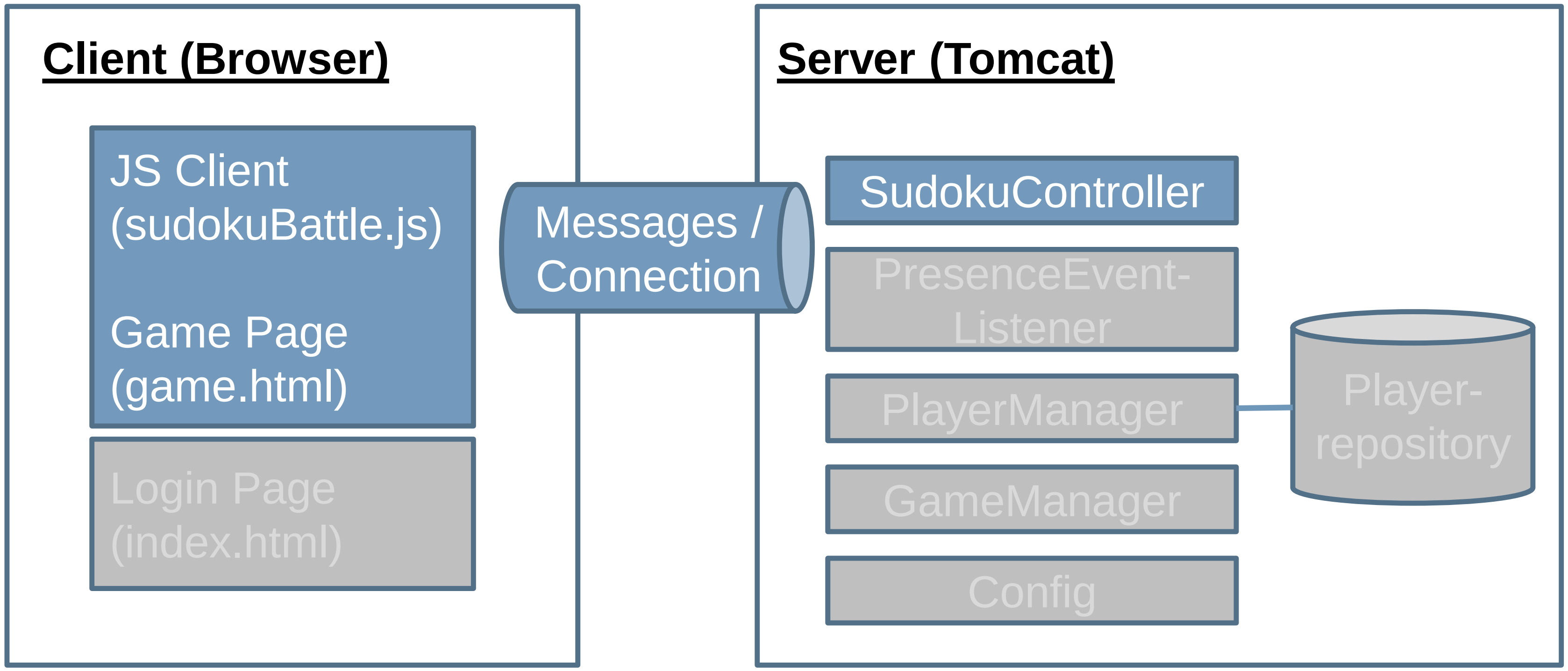
- Setup of broker for “/topic” prefixed destinations
- Sending of messages to “/topic/game/login” after new session connections
- Client: subscription to destination “/topic/game/login”

MESSAGING: BUILDING BLOCKS


lists the players that are already waiting
/app/game/players
/app/solve
send a “solve attempt” from client to server
inform all waiting players about a new player
/topic/game/login
Application (annotated Handler methods)
/topic/game/start
inform all players about the start of the game
inform all players about a board /score update
/topic/game/update
(in-memory)
/user/queue/attempt
inform a particular user about the outcome of an attempt
Clients
response channel

ELCA

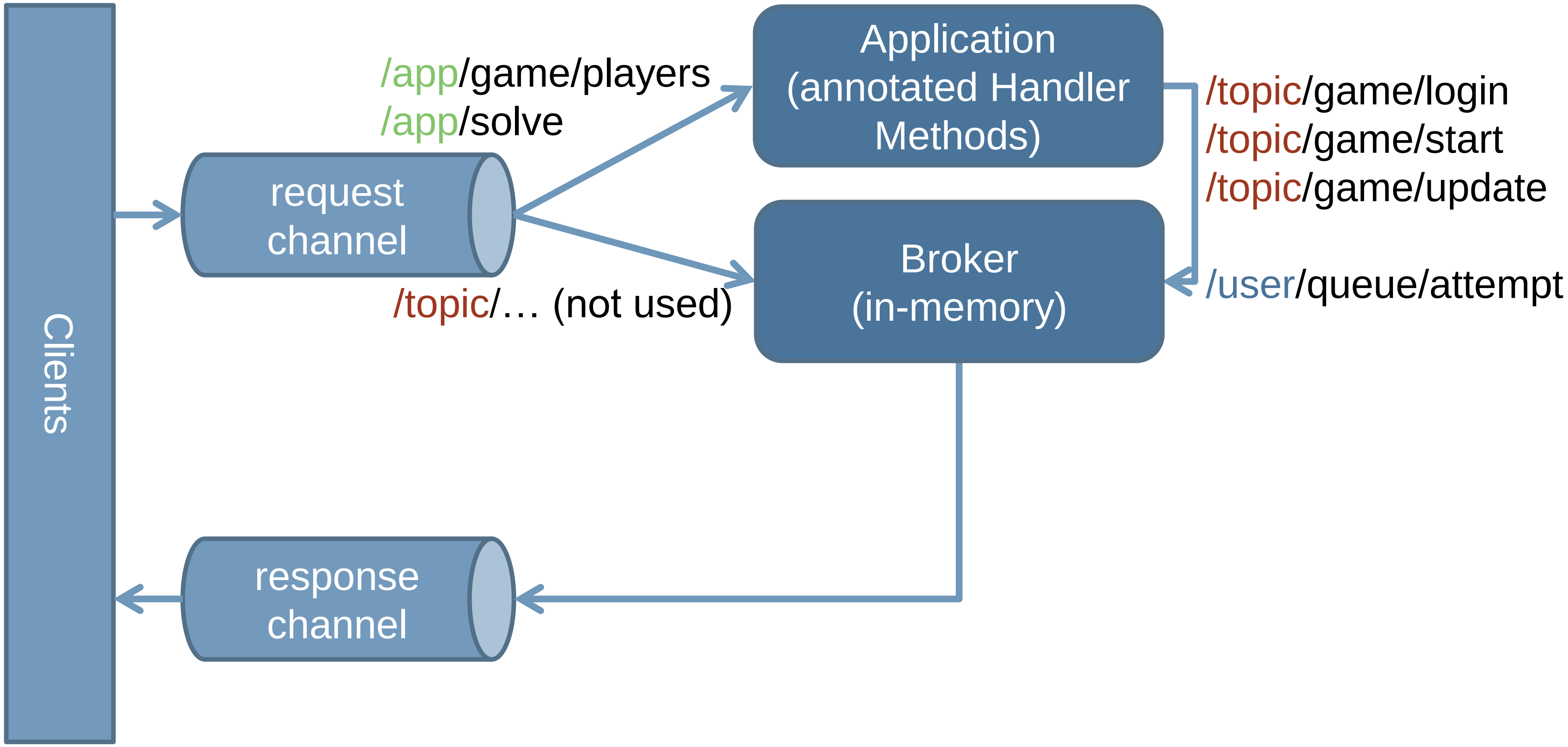
STAGE 3: SENDING MESSAGES FROM THE CLIENT



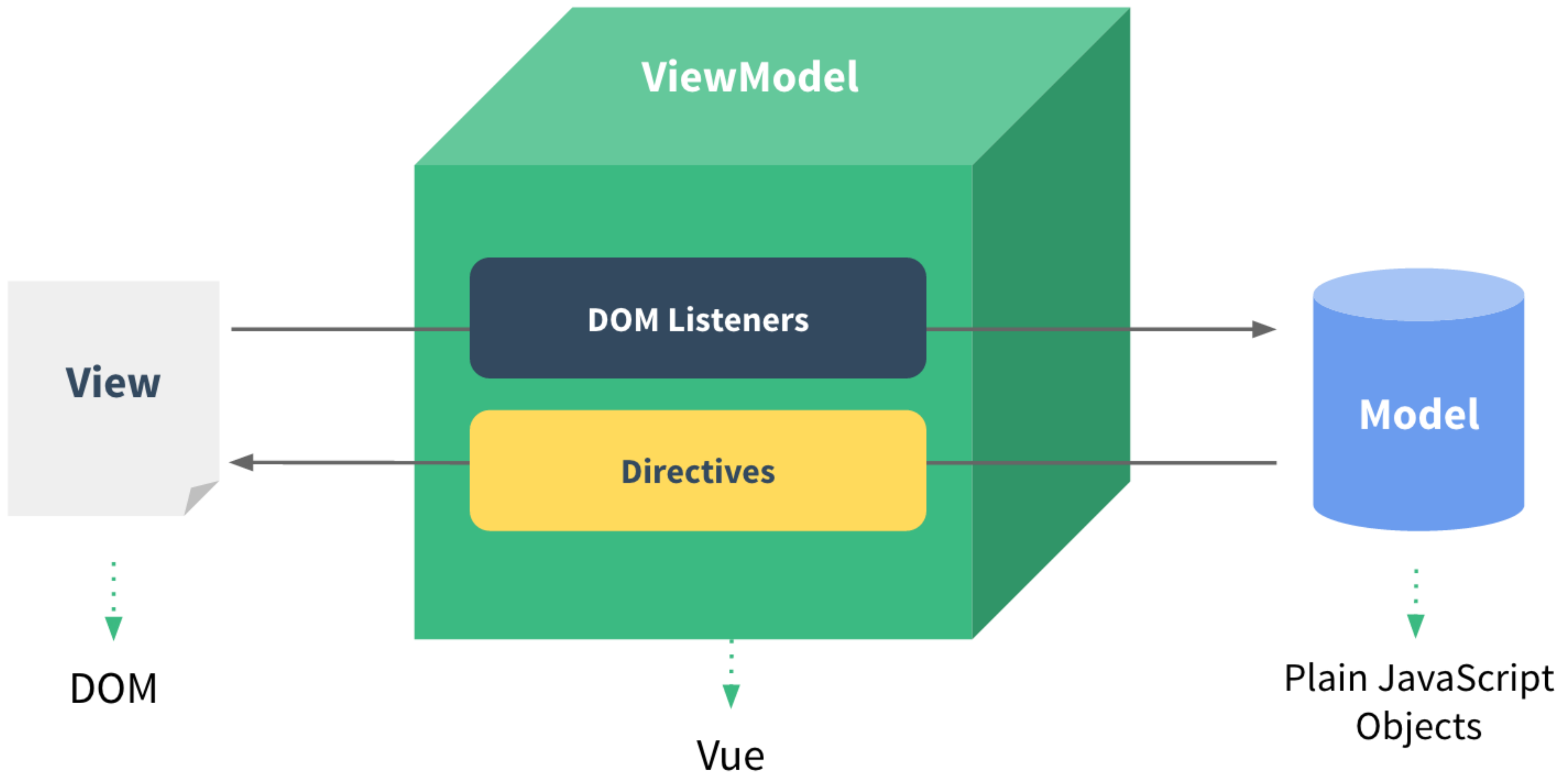
- Handling of “solve” messages on the server
  - Messages to public destination
  - Messages to personal destination
- Sending “solve messages” from the client

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 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 player with the top score wins

- **Login: http://sudokubattle.scapp.io**

URLS

- Github Repository: https://github.com/elcaIT/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	Stefan Jucker
Software Engineer	Software Engineer
martin.kempf@elca.ch	stefan.jucker@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