FitConnect

Distributed Systems and Middleware Technologies

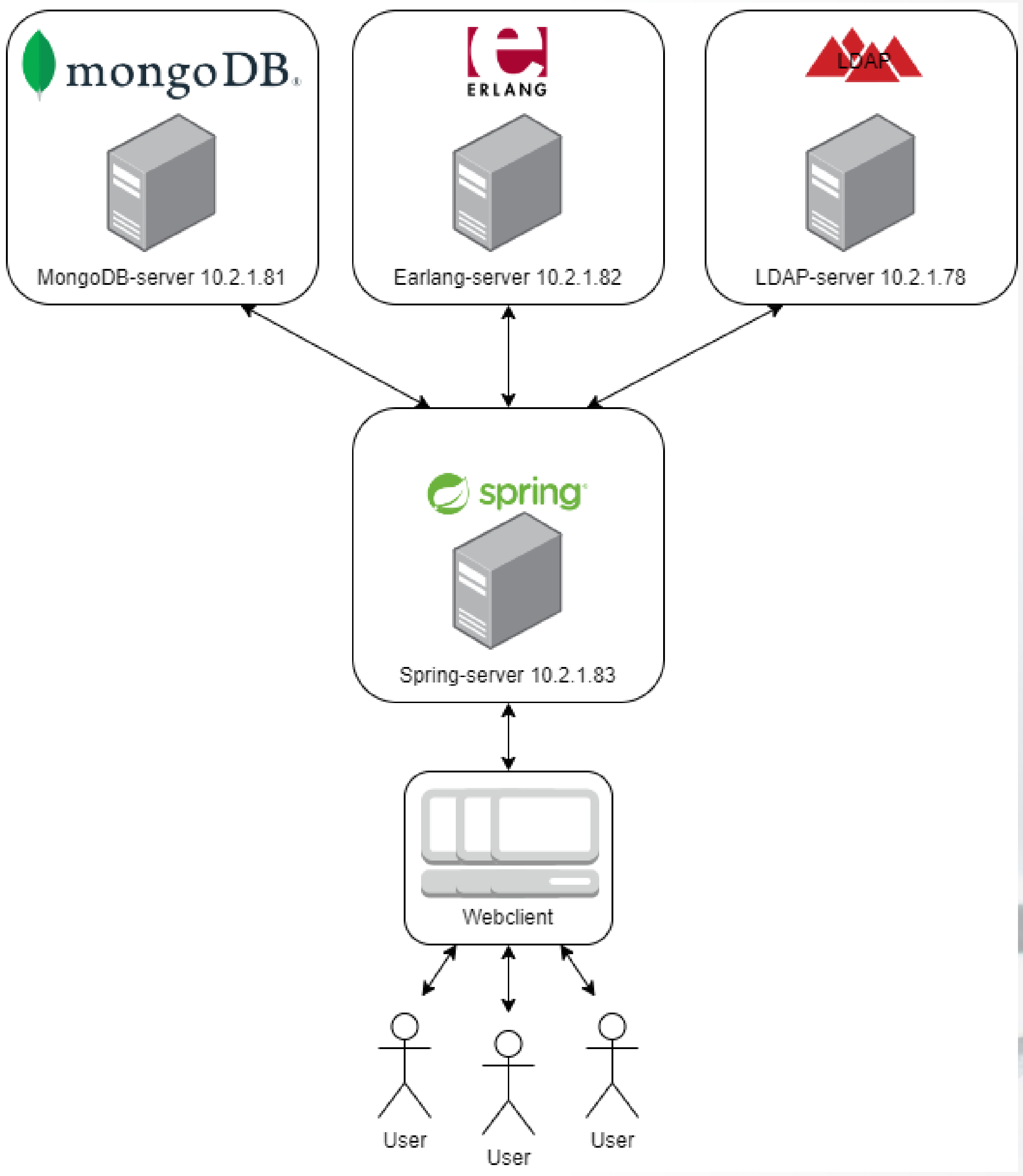
Tommaso Bertini

Giovanni Marrucci

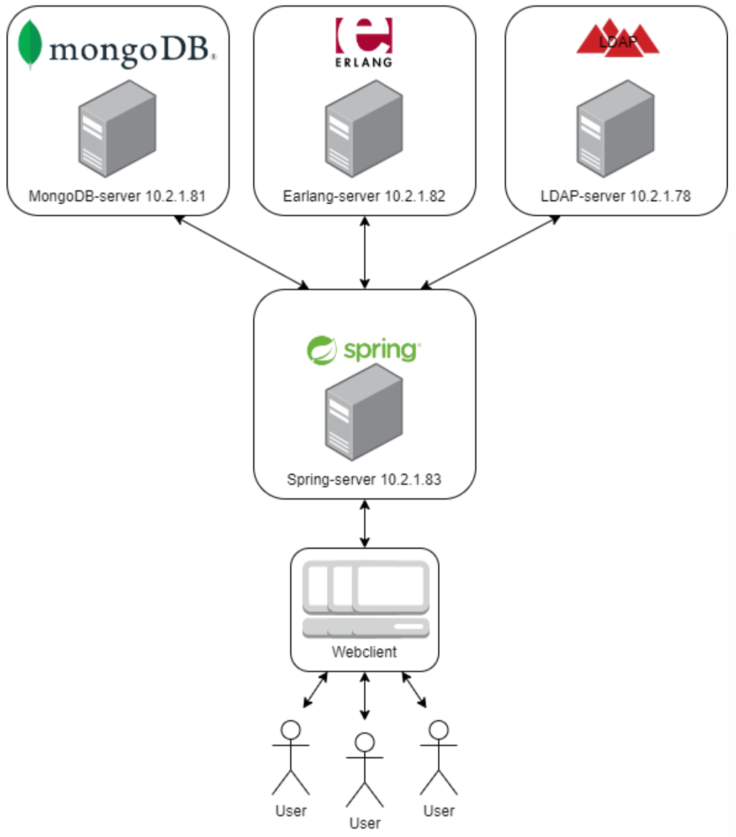
Veronica Torraca

FitConnect: specifications and requirements

- **Website** for managing fitness facility
  - Create gym courses
  - Add and book gym classes
  - Manage course and class reservations
  - Notifications system
  - Course chat
- **Concurrency** and synchronization management
- **Consistency** for users, courses and classes data stored on MongoDB and MnesiaDB



System Architecture



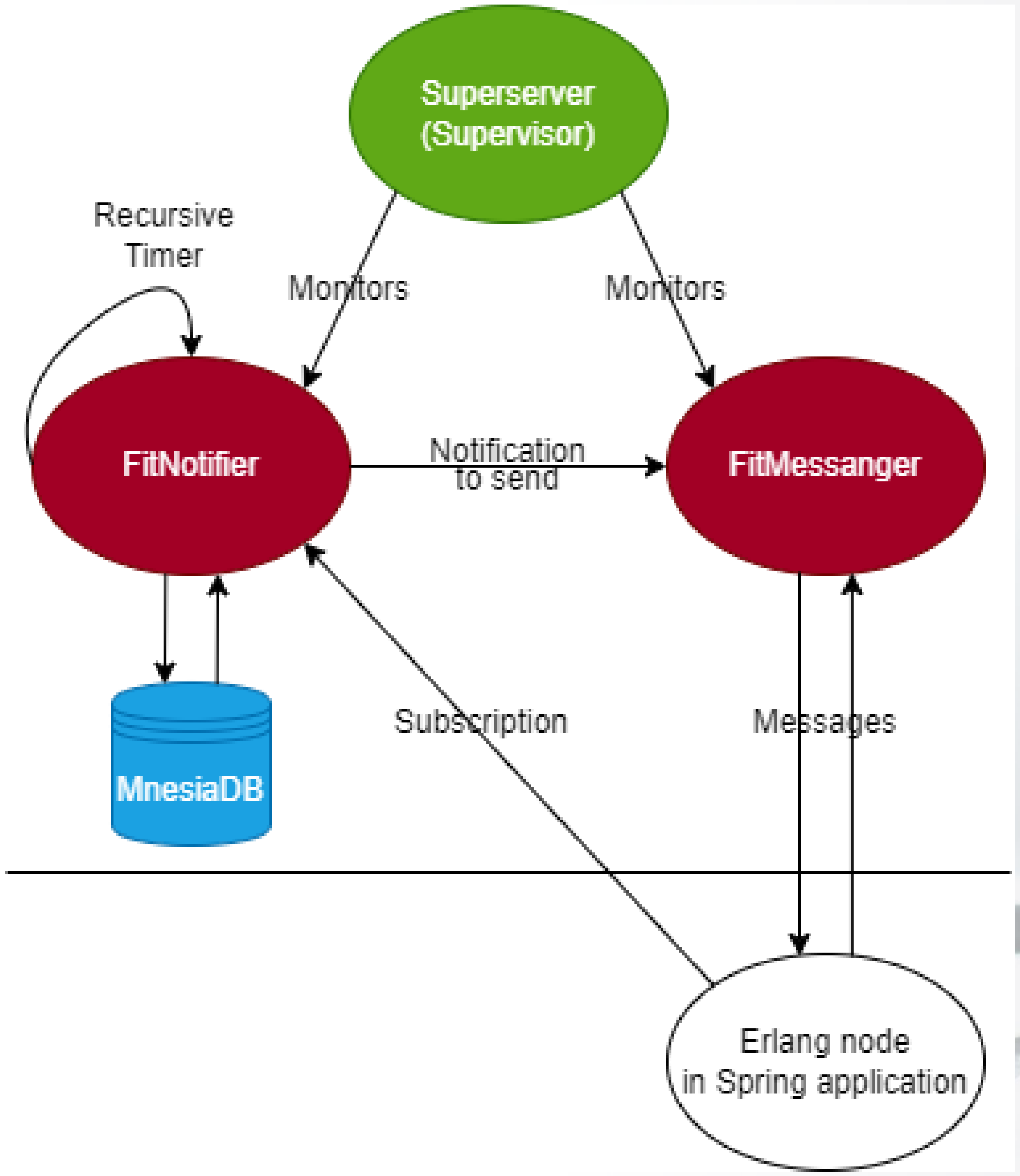
SpringBoot

- **Tomcat** webserver
- MCS pattern
- **Web interface** for client interactions
- Manages connection with **Erlang server**
- Provides support for **WebSocket**

LDAP

- Manages user **registration** and **authentication**
- Interactions via **SpringData LDAP**

Erlang


Superserver (Supervisor)
Recursive Timer
Monitors
Monitors
FitNotifier
Notification to send
FitMessanger
MnesiaDB
Subscription
Messages
Erlang node in Spring application

Erlang modules

- **Superserver**: implements supervisor behavior and monitors *FitNotifier* and *FitMessanger* using the following configuration:
  {strategy => one_for_one, intensity => 3, period => 20}
- **FitMessanger**: maintains in its state the clients connected *{“courses", "user", Pid}* and forwards notifications to them using broadcast (Message, Pids)
- **FitNotifier**: checks every 5 minutes if there are notifications to send and waits for new subscriptions from clients connected to the web server.
  It uses the **FitDb** module to perform operations on **MnesiaDB**

MongoDB

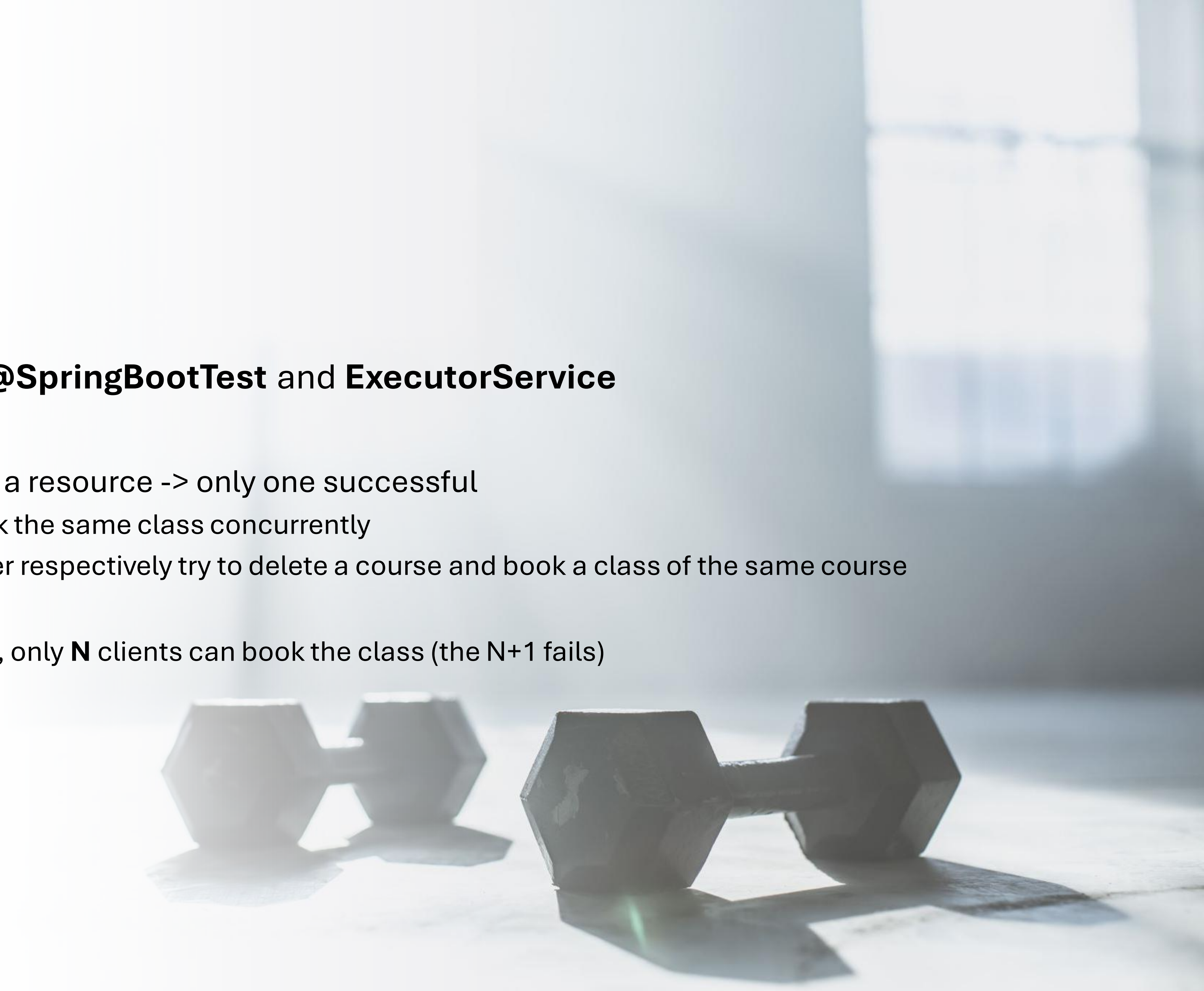
- **Stores** and **manages** persistent data
  - Via **SpringData MongoDB**
- One-to-Many and Many-to-Many relationships
- Collections
  - **Users** (trainer or client)
  - **Courses**
  - **Reservations**
  - **Messages**
- Operations
  - **Create/Delete** a course (trainer only)
  - **Add/Edit/Delete** a class schedule (trainer only)
  - **Join/Leave** a course (client only)
  - **Book/Unbook** class (client only)
  - **Browse** courses and client reservations
  - **Scheduled task** to remove past reservations documents
  - **Save/Read** chat messages

Concurrency

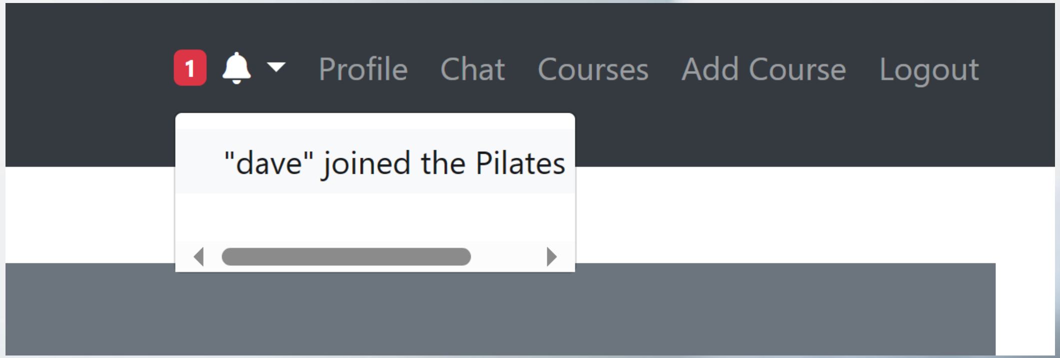
- **Limited number** of clients can book a gym class
  - No overbooking allowed
- Possible **concurrent access** to a document
  - A trainer can edit/delete a class while a client try to book it
  - Two clients try to book the same class at the same time
- Managed via **@Version** field and **Optimistic concurrency control**

Testing

- Concurrency tested via **@SpringBootTest** and **ExecutorService**
- Tests performed
  - **Concurrent access** to a resource -> only one successful
    - Two users try to book the same class concurrently
    - The trainer and a user respectively try to delete a course and book a class of the same course
  - **Overbooking**
    - If **N** places available, only **N** clients can book the class (the N+1 fails)



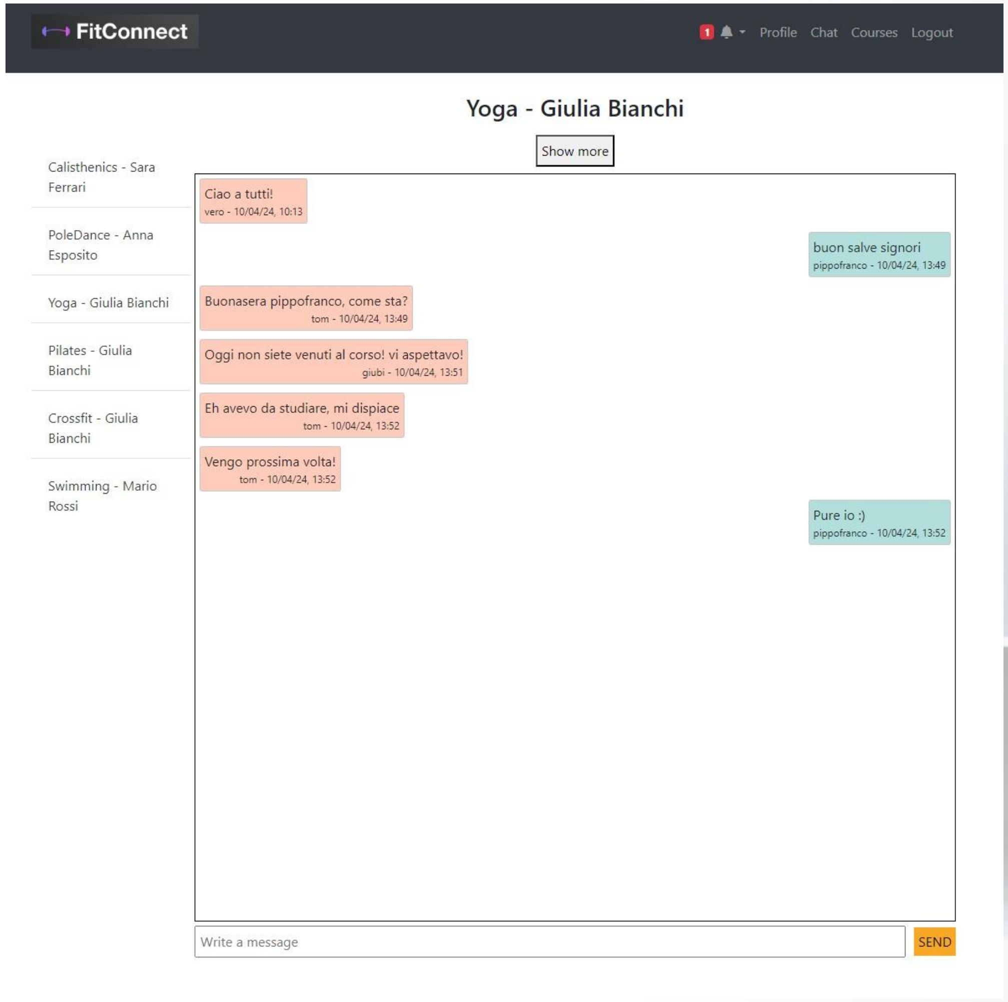
Notification system



"dave" joined the Pilates

- Implemented using **Erlang** and **Web Socket**
- Notification **sent to all booked users**
  - Half an hour **before the start** of the booked class
  - Every time a **class** schedule is **modified**
- Notification sent to all members of a course (both trainer or client)
  - When a new client **joins** the course
  - When a user **leaves** the course
  - When the trainer **delete** the course

Group Chat



Yoga - Giulia Bianchi

Ciao a tutti!
vero - 10/04/24, 10:13

buon salve signori
pippofranco - 10/04/24, 13:49

Buonasera pippofranco, come sta?
tom - 10/04/24, 13:49

Oggi non siete venuti al corso! vi aspettavo!
giubi - 10/04/24, 13:51

Eh avevo da studiare, mi dispiace
tom - 10/04/24, 13:52

Vengo prossima volta!
tom - 10/04/24, 13:52

Pure io :)
pippofranco - 10/04/24, 13:52

- Implemented using **STOMP protocol** over **Web Socket**
- A Chat for each Course
- All clients enrolled in a course can participate to the related group chat
- JavaScript frontend
- Java middleware