MYISSUES

A Software Engineering Project

CONTENTS

- The project
  - The teams
  - The process (SCRUM)
- The product
  - Sprint 1
  - Experts committee
    - JDBC
    - SOCKETS

CONTENTS

THE TEAMS

https://gitalcoi.dsic.upv.es/jesparza/proyecto-iso/-/wikis/teams

THE TEAMS

Ms Teams

- Each team creates one team in Ms Teams, with the name “ISO_<Team name>”
- All members of the team are added
- You must add me to the team

THE TEAMS

Gitlab

- Each team creates one project in Gitlab with the name “MyIssues”
- All team members must be added with all privileges
- The Git repo is essential
  - git:/doc → design docs, scrum docs, …
  - git:/src → source code
  - git:/lib → dependencies, libraries, ...

CONTENTS

THE PROCESS

- SCRUM
- Backlog with issues

THE PROCESS

- 4 SPRINTS
- https://gitalcoi.dsic.upv.es/jesparza/proyecto-iso/-/wikis/home

DATES	TODO TASKS	DELIVERABLES
10/06/21 - 10/19/21	Join a team	-
10/20/21 - 11/02/21	SCRUM Sprint 1	Due date: 11/02/21 • Software version 1 • Weekly reports • Sprint review presentation
11/03/21 - 11/16/21	SCRUM Sprint 2	Due date: 11/16/21 • Software version 2 • Weekly reports • Sprint review presentation
11/17/21 - 11/30/21	SCRUM Sprint 3	Due date: 11/30/21 • Software version 3 • Weekly reports • Sprint review presentation
12/01/21 - 12/22/21	SCRUM Sprint 4	Due date: 12/22/21 • Software version 4 • Weekly reports • Final review presentation

THE PROCESS

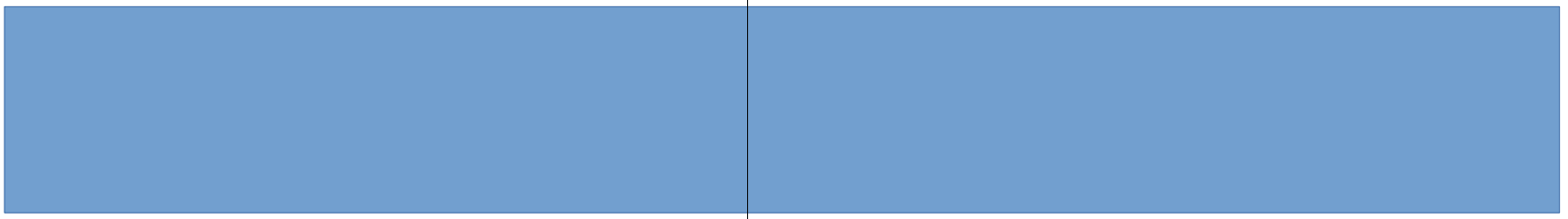
- For each sprint (2-3-4 weeks long)
  - Talk to Mr. Roberino/me to fix product/sprint backlog (backlog with issues)
  - Each team self-organizes and assigns tasks to the team members (Gitlab)
  - Each team plans 2 meetings/week (Ms Teams)
  - Each member records his activity (Gitlab)
  - Each member uploads the code (Gitlab)

THE PROCESS

- For each sprint. Results (evidences)
  - Meeting reports: min 2 reports/week, under git:/docs/scrum/sprint[1-4]/meeting[1-2].pdf
  - Sprint review: report under git:/docs/scrum/sprint[1-4]/review.pdf
  - Presentation of results: 5’ long, show product working

THE PROCESS

- In each lab session



- Sprint review: presentation of sprint results
- Demo of product working

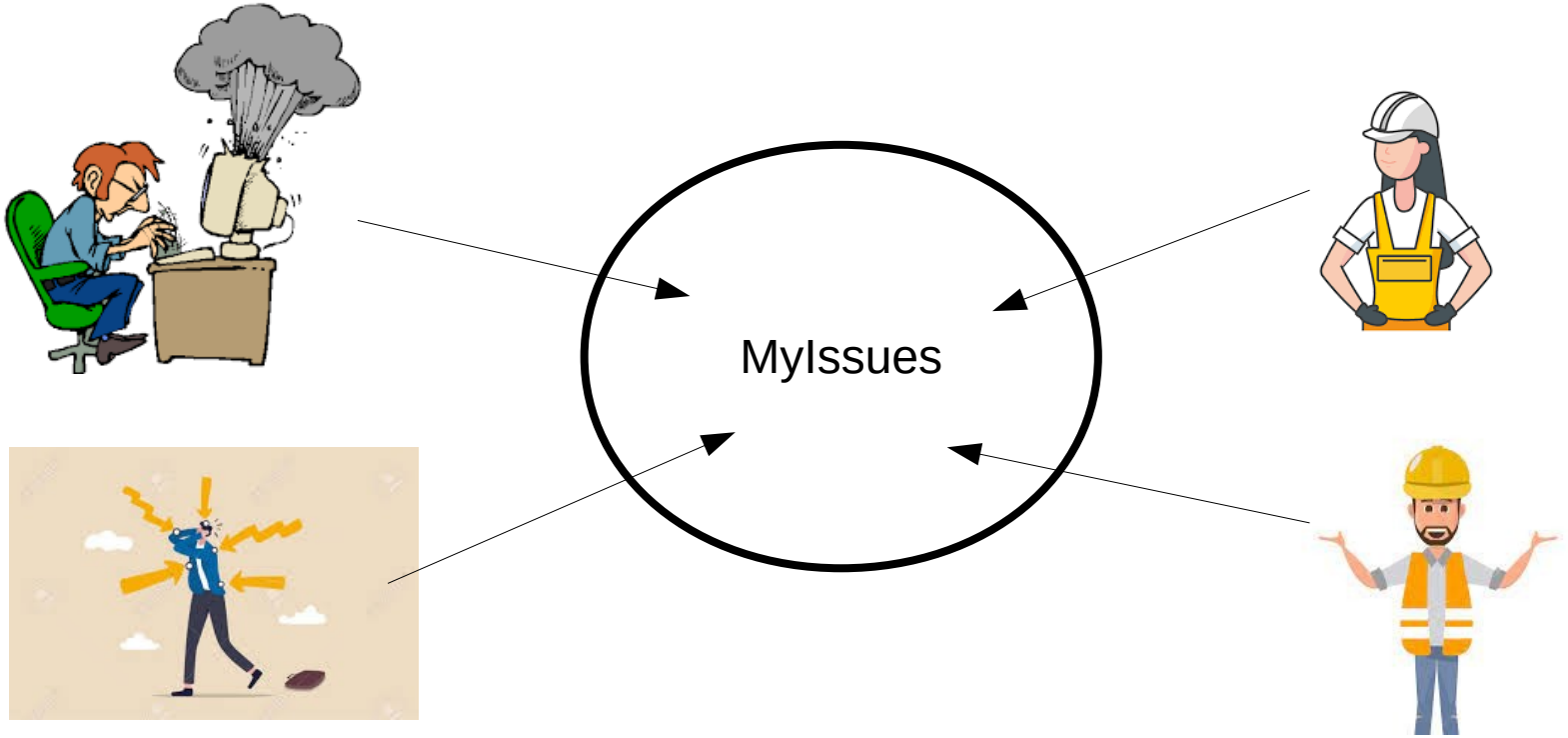
1. Talk to Mr Roberino
2. Fix next sprint backlog
3. Experts committee: discuss technical details

CONTENTS

- The project
  - The teams
  - The process (SCRUM)
- The product
  - Sprint 1
  - Experts committee
    - JDBC
    - SOCKETS

THE PRODUCT

- Keep track of issues: MYISSUES
- Users register problems. Users solve problems.



THE PRODUCT

- We will develop the product in 4 sprints
- Each sprint includes new requirements

CONTENTS

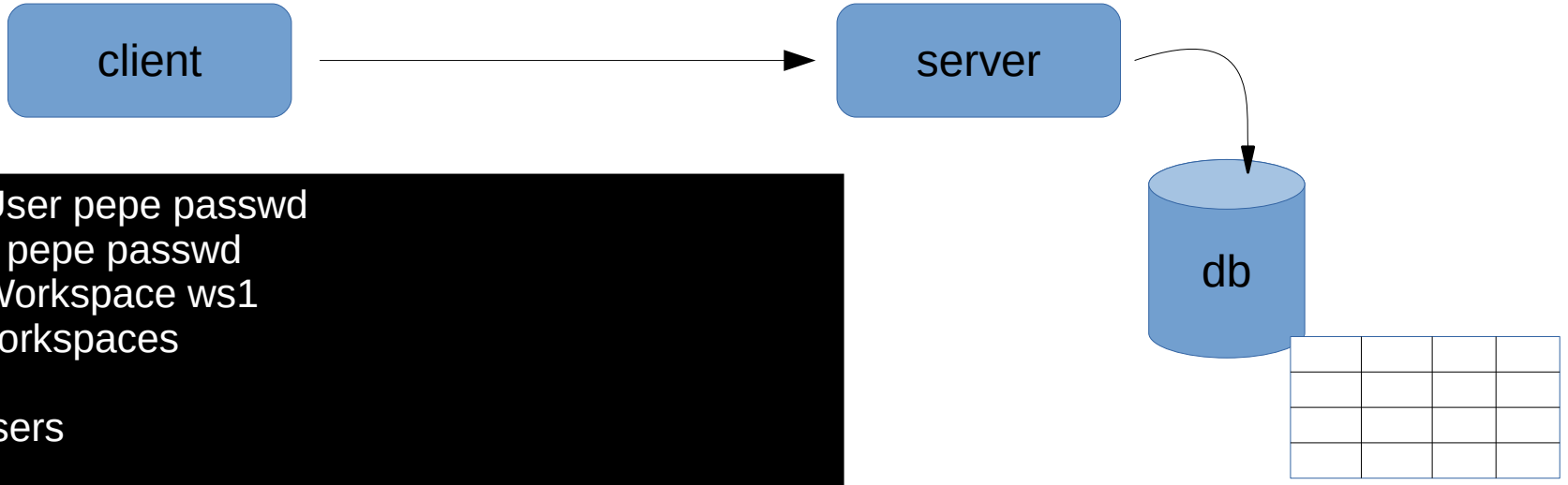
- The project
  - The teams
  - The process (SCRUM)
- The product
  - Sprint 1
  - Experts committee
    - JDBC
    - SOCKETS

SPRINT 1

- Let’s begin first sprint!!! - Mr Roberino

SPRINT 1

Conclusions




```
> addUser pepe passwd
> login pepe passwd
> addWorkspace ws1
> listWorkspaces
ws1
> listUsers
juan
> join juan ws1
> listMembers ws1
pepe juan
> leave pepe ws1
> listWorkspaces
(empty)
```

SPRINT 1

Conclusions

- Users/workspaces management
- Server:
  - Define data model (database tables, etc.)
  - Implement remote API
- Client:
  - Command line: login, logout, addUser, removeUser, listUsers, addWorkspace, removeWorkspace, listWorkspaces, join, leave

CONTENTS

- The project
  - The teams
  - The process (SCRUM)
- The product
  - Sprint 1
  - Experts committee
    - JDBC
    - SOCKETS

EXPERTS COMMITTEE

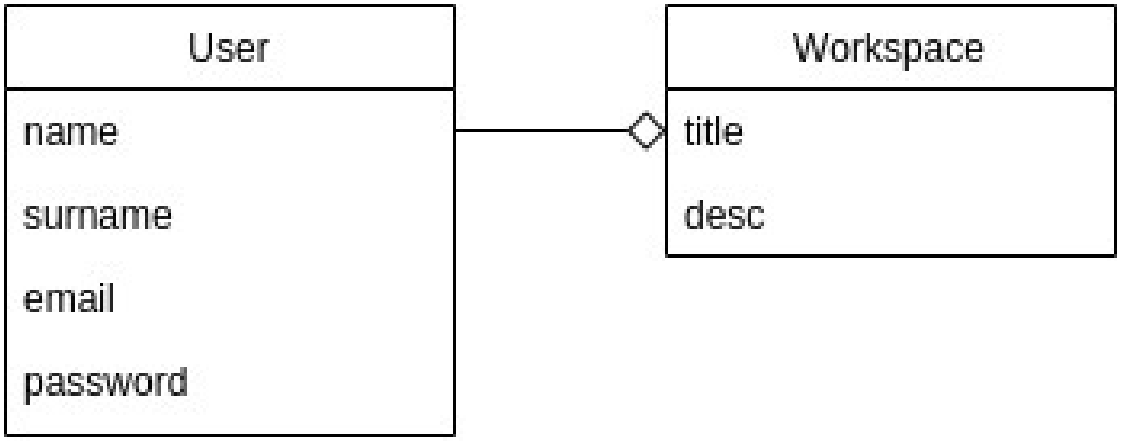
- Databases with JDBC
- Communications with sockets

CONTENTS

- The project
  - The teams
  - The process (SCRUM)
- The product
  - Sprint 1
  - Experts committee
    - JDBC
    - SOCKETS

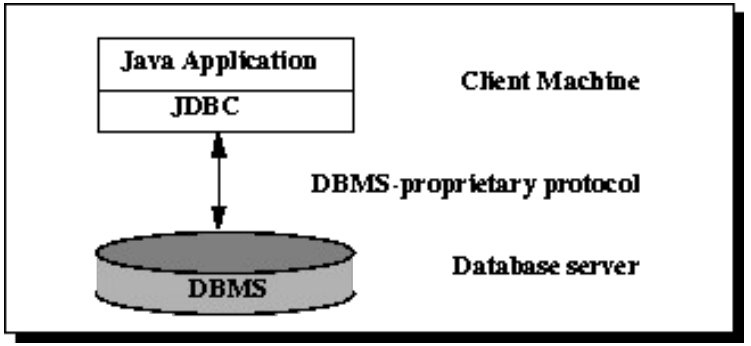
JDBC

- Data model
- User/workspace



JDBC

- Java Database Connectivity
- Common API for any database (java.sql.*)
- We need a driver which translates calls



- Let’s use SQLite

JDBC

- We add driver (xxx.jar) to the project CLASSPATH
- We always follow the same steps:
  1) Open connection to database
  2) Send queries and updates to database
  3) Close connection

JDBC. Open connection

```
Connection con = DriverManager.getConnection(
                    "jdbc:sqlite:sample.db",
                    username,
                    password);
```

JDBC. Updates

```
Statement stmt = con.createStatement();

stmt.executeUpdate("CREATE TABLE msgs(id INT, content TEXT)");

int count = stmt.executeUpdate("INSERT INTO msgs VALUES (1,
'hello')");
```

JDBC. Query

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM msgs");
while (rs.next()) {
  int x = rs.getInt(1);
  String s = rs.getString(2);
}
```

JDBC. Close connection

```
con.close();
```

SQLITE CLI

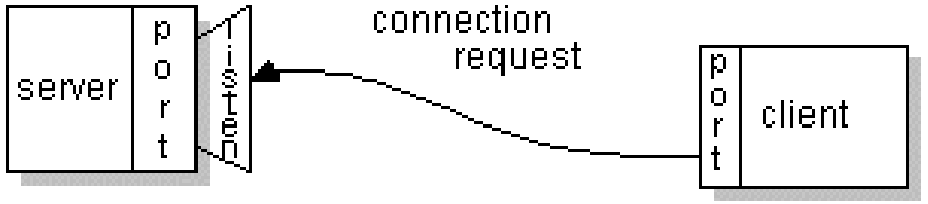
```
> sqlite3 sample.db
> .help
> .tables
> .schema msgs
> select * from msgs;
> insert into msgs values (2, ‘pepe’);
> select * from msgs;
```

CONTENTS

- The project
  - The teams
  - The process (SCRUM)
- The product
  - Sprint 1
  - Experts committee
    - JDBC
    - SOCKETS

SOCKETS

- Java networking (client/server) in java.net.*
- A server socket listens to client requests



- If the request is accepted a connections gets created

SOCKETS

- We always follow the same steps
- On the server
  1) We create the server socket
  2) We accept client input connections
  3) We read/write to client
  4) Close
- On the client
  1) We create client socket
  2) We read/write to server

SOCKETS. SERVER SOCKET

```
try {
  ServerSocket serverSocket = new ServerSocket(portNumber);
  Socket clientSocket = serverSocket.accept();
  BufferedReader in = new BufferedReader(
        new InputStreamReader(clientSocket.getInputStream()));
  PrintWriter out =
        new PrintWriter(clientSocket.getOutputStream(), true);
  out.println(in.readLine());
  clientSocket.close();
  serverSocket.close();
} catch (Exception e) { ... }
```

SOCKETS. CLIENT SOCKET

```
try {
  Socket clientSocket = new Socket(hostName, portNumber);
  PrintWriter out = new PrintWriter(clientSocket.getOutputStream());
  BufferedReader in = new BufferedReader(
        new InputStreamReader(clientSocket.getInputStream()));
  out.println("hello");
  out.flush();
  String msg = in.readLine();
  clientSocket.close();
} catch (Exception e) { ... }
```