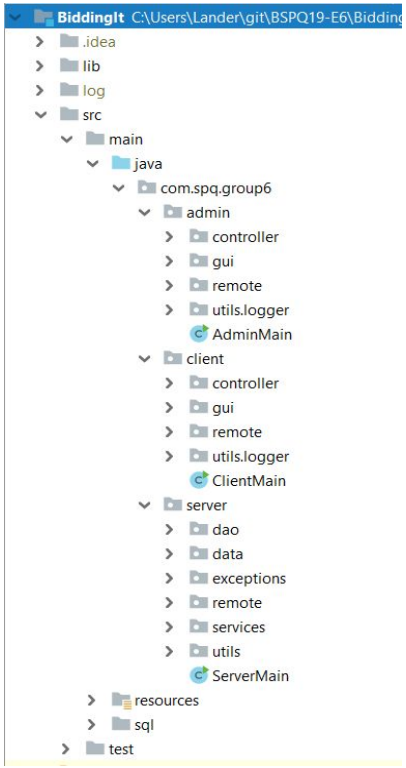
BSPQ19-E6

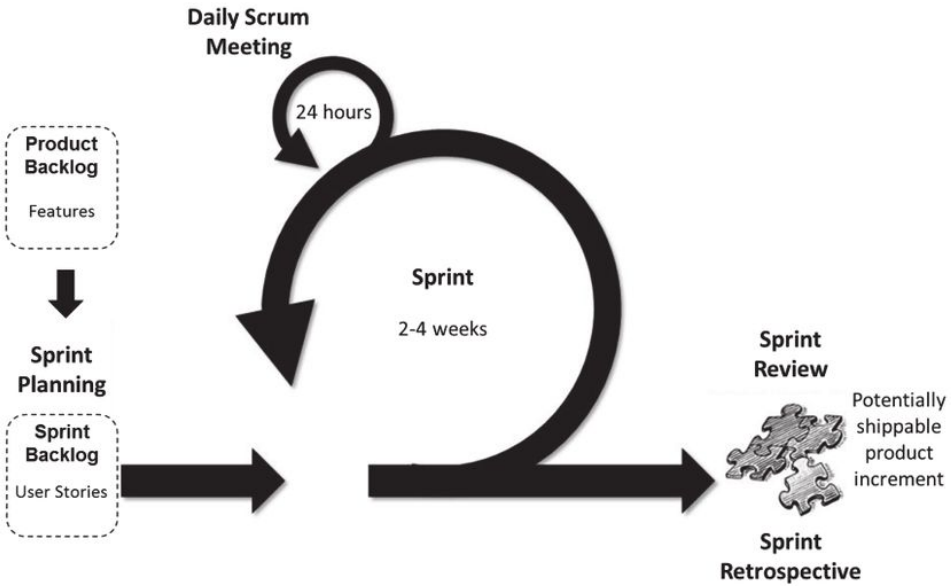
SCRUM Post-Game Phase

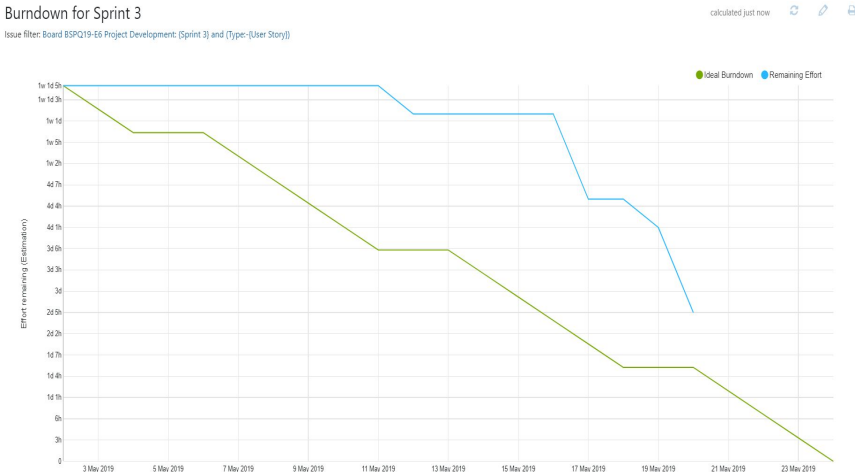
Purpose of the Application and its Architecture

- International bidding application.
  - Be able to sell my products.
  - Be able to bid for products.
  - Support for international users.
- Simple and intuitive GUI.
- Client-server structure.
  - Two client-types: User and Administrator
- Make a use of the tools and services explained in class for the correct development of the application.
  - Youtrack, Github, Maven, Jenkins, Doxygen, ...



Brief description of the process using SCRUM

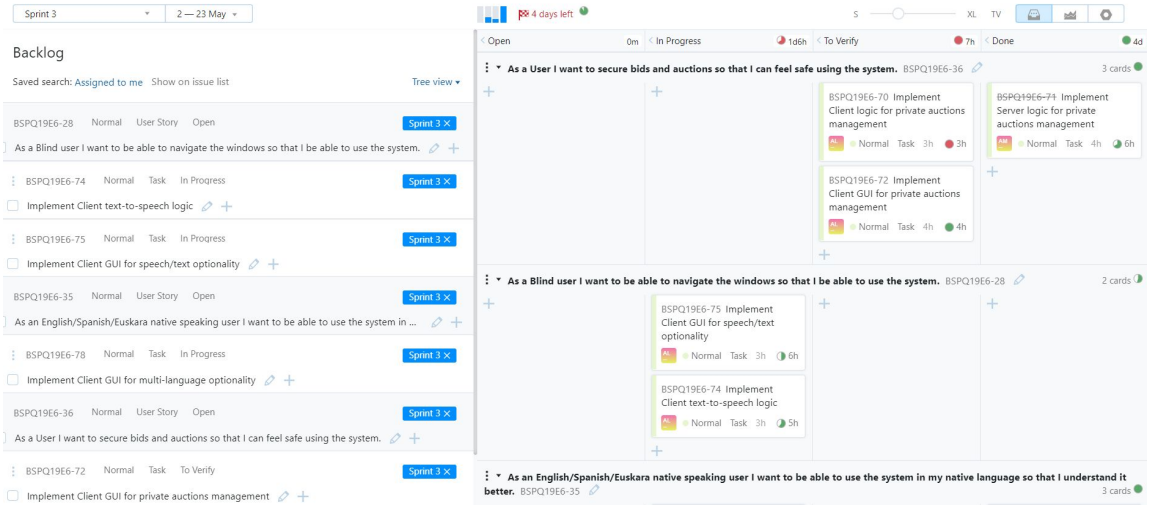


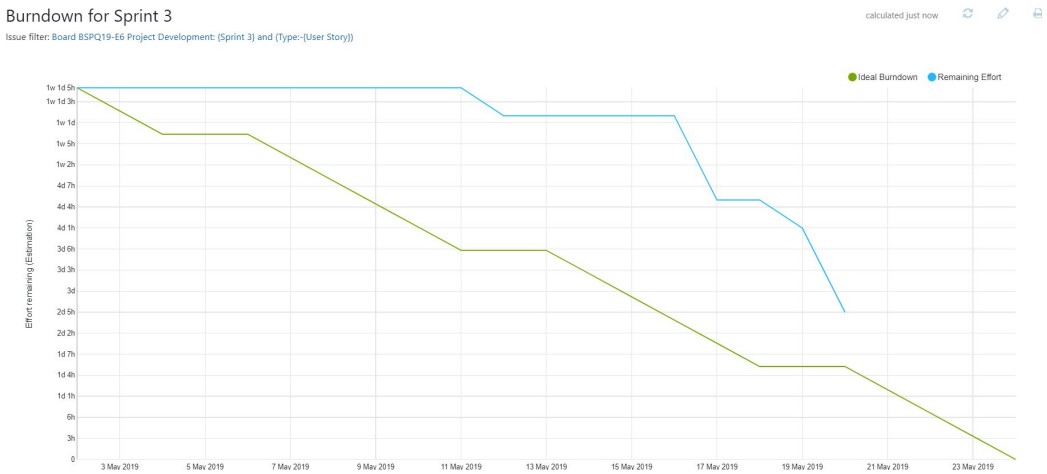


Description of the use of YouTrackfor for Scrum and the integration with GitHub

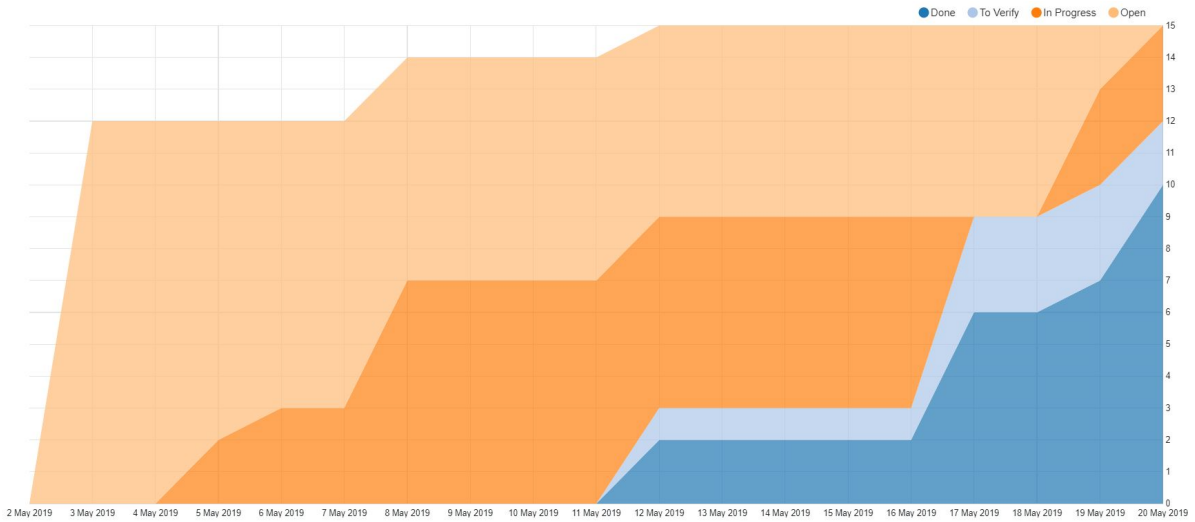
- A Product Backlog was created (User stories).
- The user stories were sorted in 3 sprints in order of importance.
- For each sprint:
  - These User Stories were “divided” into Tasks and their time was estimated.
  - The tasks were distributed among the team members taking into account personal preference and knowledge.
  - Dependence between GitHub and YouTrack is created.
  - Therefore it’s easier to Push & Commit Tasks, which will be directly updated in YouTrack as well (#Task, #Duration, ...)

Sprint 3


Sprint 3
2 — 23 May
4 days left
Backlog
Saved search: Assigned to me
Show on issue list
Tree view
BSPQ19E6-28 Normal User Story Open
As a Blind user I want to be able to navigate the windows so that I be able to use the system.
BSPQ19E6-74 Normal Task In Progress
Implement Client text-to-speech logic
BSPQ19E6-75 Normal Task In Progress
Implement Client GUI for speech/text optionality
BSPQ19E6-35 Normal User Story Open
As an English/Spanish/Euskara native speaking user I want to be able to use the system in ...
BSPQ19E6-78 Normal Task In Progress
Implement Client GUI for multi-language optionality
BSPQ19E6-36 Normal User Story Open
As a User I want to secure bids and auctions so that I can feel safe using the system.
BSPQ19E6-72 Normal Task To Verify
Implement Client GUI for private auctions management
Open
In Progress
To Verify
Done
As a User I want to secure bids and auctions so that I can feel safe using the system. BSPQ19E6-36
BSPQ19E6-70 Implement Client logic for private auctions management
BSPQ19E6-71 Implement Server logic for private auctions management
BSPQ19E6-72 Implement Client GUI for private auctions management
As a Blind user I want to be able to navigate the windows so that I be able to use the system. BSPQ19E6-28
BSPQ19E6-75 Implement Client GUI for speech/text optionality
BSPQ19E6-74 Implement Client text-to-speech logic
As an English/Spanish/Euskara native speaking user I want to be able to use the system in my native language so that I understand it better. BSPQ19E6-35


Burndown for Sprint 3
calculated just now
Issue filter: Board BSPQ19-E6 Project Development: (Sprint 3) and (Type:-(User Story))
Ideal Burndown
Remaining Effort
Effort remaining (Estimation)
1w 1d 5h
1w 1d 3h
1w 1d
1w 5h
1w 2h
4d 7h
4d 4h
4d 1h
3d 6h
3d 3h
3d
2d 5h
2d 2h
1d 7h
1d 4h
1d 1h
6h
3h
0
3 May 2019
5 May 2019
7 May 2019
9 May 2019
11 May 2019
13 May 2019
15 May 2019
17 May 2019
19 May 2019
21 May 2019
23 May 2019

Sprint 3



Aratz Manterola ●

3 May 2019 | **2h 00m** | Development | Added Travis to Github

Real Time: ? ⟶ 2h

Aratz Manterola ● committed changes 3 May 2019 19:40

Travis file from previous commit (forgot to add it) 661f2cbf

3 May 2019 19:40

Added Travis corrected file and ClientController fixed tests aaf6fb82

3 May 2019 19:41

Create .travis.yml again 7e686364

3 May 2019 19:41

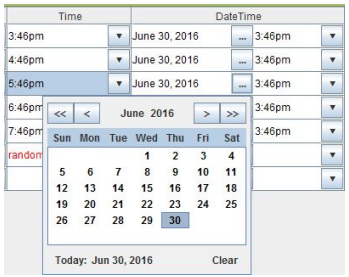
Test for Travis c955b509

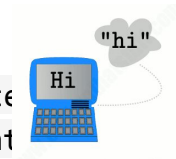
Aratz Manterola ● 4 May 2019 18:43

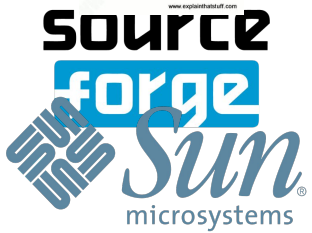
State: Open ⟶ In Progress

Description of the use of Maven

- POM File containing the project details, properties, dependencies, build, profiles and repositories. (430 lines approx.)
- Dependencies of other Tools Libraries can be assigned -> same build.
- Some 3rd party jars were installed using `mvn install` and then modifying the dependencies (such as LGoodDatePicker and freeTTS (10+ .jar)).
- Important Prompts:
  - `mvn clean compile (compile project)`
  - `mvn datanucleus:schema-create (create DB) :schema-delete (delete`
  - `mvn exec:java -Pserver (execute Server) -Pclient (execute Client -Padmin (execute Admin client)`
  - `mvn test (will generate the Contiperf File)`
  - `mvn doxygen:report (to generate Documentation)`
  - `mvn test jacoco:check (coverage)`





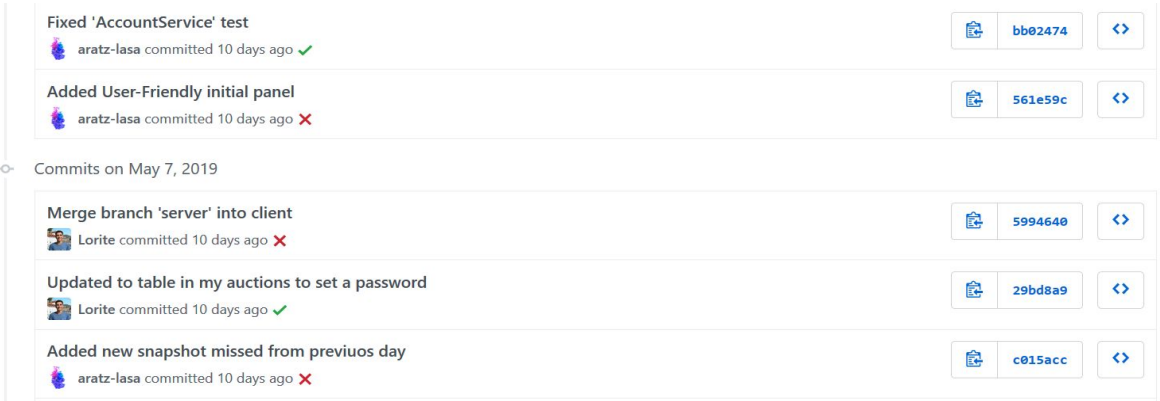


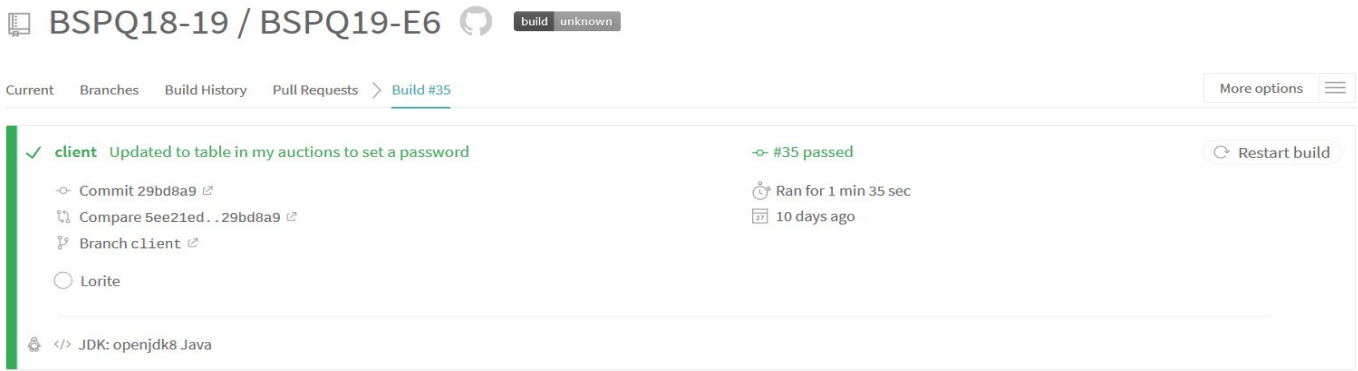
About Unit, Coverage and Performance Testing

- Separate tests for **performance** (Contiperf)
  - With concurrency, we could not clean-up after each test
  - ClientControllerTest vs ConcurrentClientControllerTest
- **Coverage** excluded folders (Jacoco)
  - gui,ClientRemoteObserver, data, exceptions, logger, AdminMain, ClientMain, ServerMain
- **Coverage** Evolution - snapshots
  - Sprint 2: 66% - 45%
  - Sprint 3: 81% - 75%
- **Unit** testing (JUnit)
  - Few edge cases (no null-checkers). For example: ‘Bid’
  - Mockito-like ‘FakeRemoteObserver’

Use of Jenkins and screenshot showing its use

- Travis
- Github integration
- .travis.yml
- Travis dashboard





Use of Doxygen and SLF4j. GitHub Pages.

- Our project is fully done with Log4j, which is a library that allows filtering messages according to their importance

ClientLogger.java

```java
package com.spq.group6.client.utils.logger;

import org.apache.log4j.Logger;

public class ClientLogger {
    public static Logger logger = Logger.getLogger(ClientLogger.class);
}
```

- We also use Doxigen which is a plug-in based in LateX to document the source code with “mvn doxygen:report”
  - Most of Server documented (interfaces, not their implementation)
  - For now, only 3 classes are “fully” documented with Doxygen style

- Our documentation will not be available in GitHub Pages until the project is definitely finished and we can commit on branch master.

```java
/**
 * Method for bidding an Auction
 * <p>
 * Whenever a User wants to Bid in an Auction,
 * this method is called.
 * As every Bid is related to a User and an Auction,
 * it is necessary to pass the User and the Auction as arguments.
 *
 * @param auction Auction will be bidded to
 * @param user    Bid creator User
 * @param amount  Bidded amount (money)
 * @return Auction with the new Bid
 * @throws RemoteException  is raised in case of Error on RMI connection
 * @throws AuctionException is raised in case of invalid Bid amount
 */
Auction bid(Auction auction, User user, float amount) throws RemoteException, AuctionException;
```

Example of documentation done for Doxygen

Events and Difficulties

- Difficulties on setting up the environment:
  - First time using Maven.
  - Using github correctly (first time resolving merges)
- Difficulties during the sprints:
  - Manage to get everything done on time at second sprint because we estimated badly the needed effort
  - Concurrent clients → locks for every Auction and User
    - Avoid deadlock
  - JDO not made for concurrency (also implemented global lock)
  - Some basic classes (like ArrayLists) not made for RMI
    - Do not implement Serializable, so they are not correctly marshalled

Demo


BiddingIt client
BiddingIt
Welcome to the best bidding system in the world!
My auctions
Market
My products
Account
Log out
chrome