Professional Developer's Tool

Distributed Outsourced Software Engineering 2014

The teams

- Requirement: Kazan5
  - Munir Makhmutov (team leader)
  - Maxim Ibragimov
  - Farida Makhmutova
- Front-End: Madrid2
  - Williams Aguilera (team leader)
  - Rasmus Pries-Heje
  - Javier Alonso
- Back-End: Milan3
  - Mikel Vuka
  - Federico Reghenzani
  - Filippo Pagano

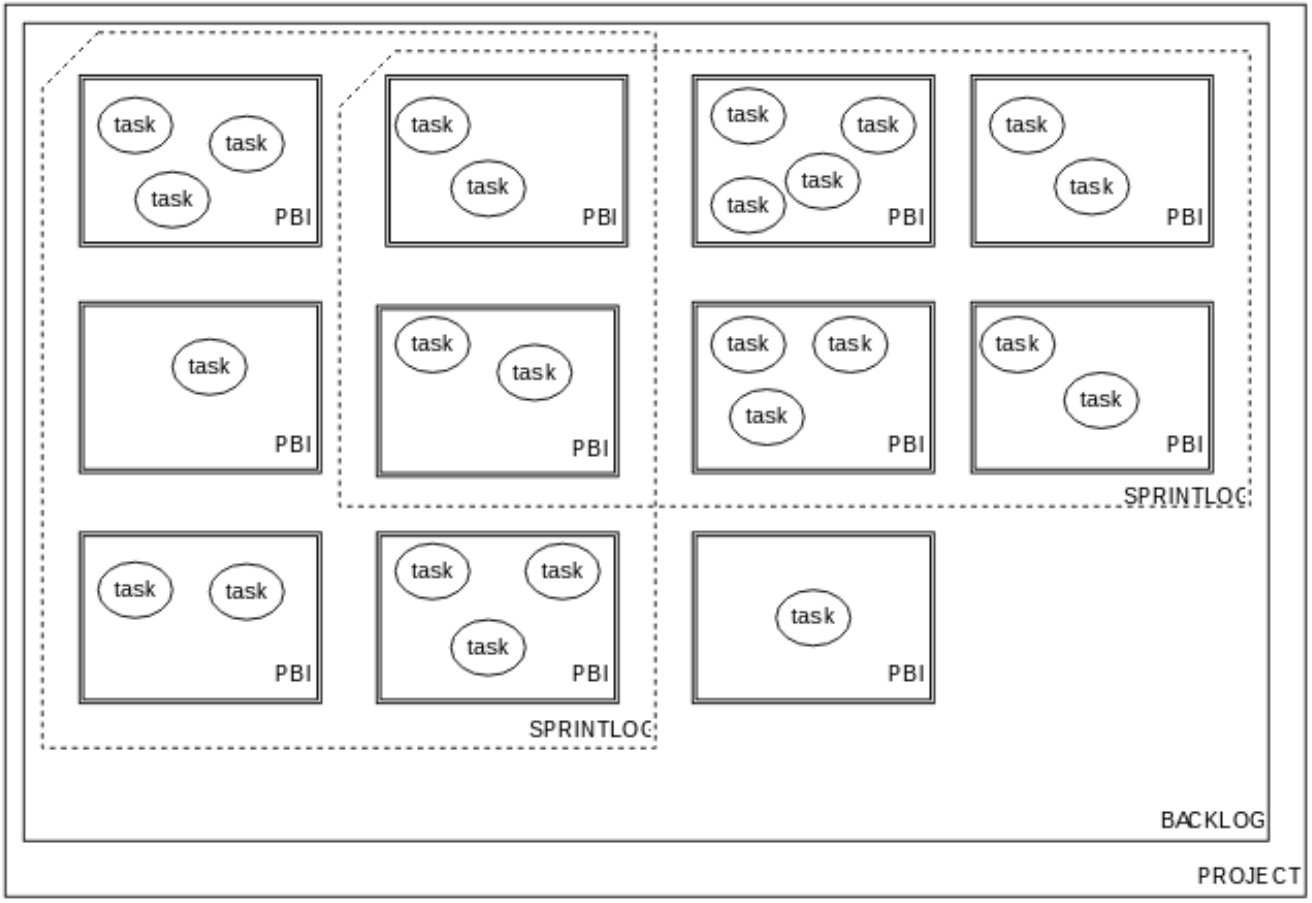
Professional Developer's Tool

From SRS document:

"Professional Developer's Tool should help to support distributed software development.

It should provide creating and deleting accounts, communication tools for members, possibility of assigning tasks to developers, controlling of the project execution, tracking progress, statistics of developers' work and developers chart. "

Components



Actors

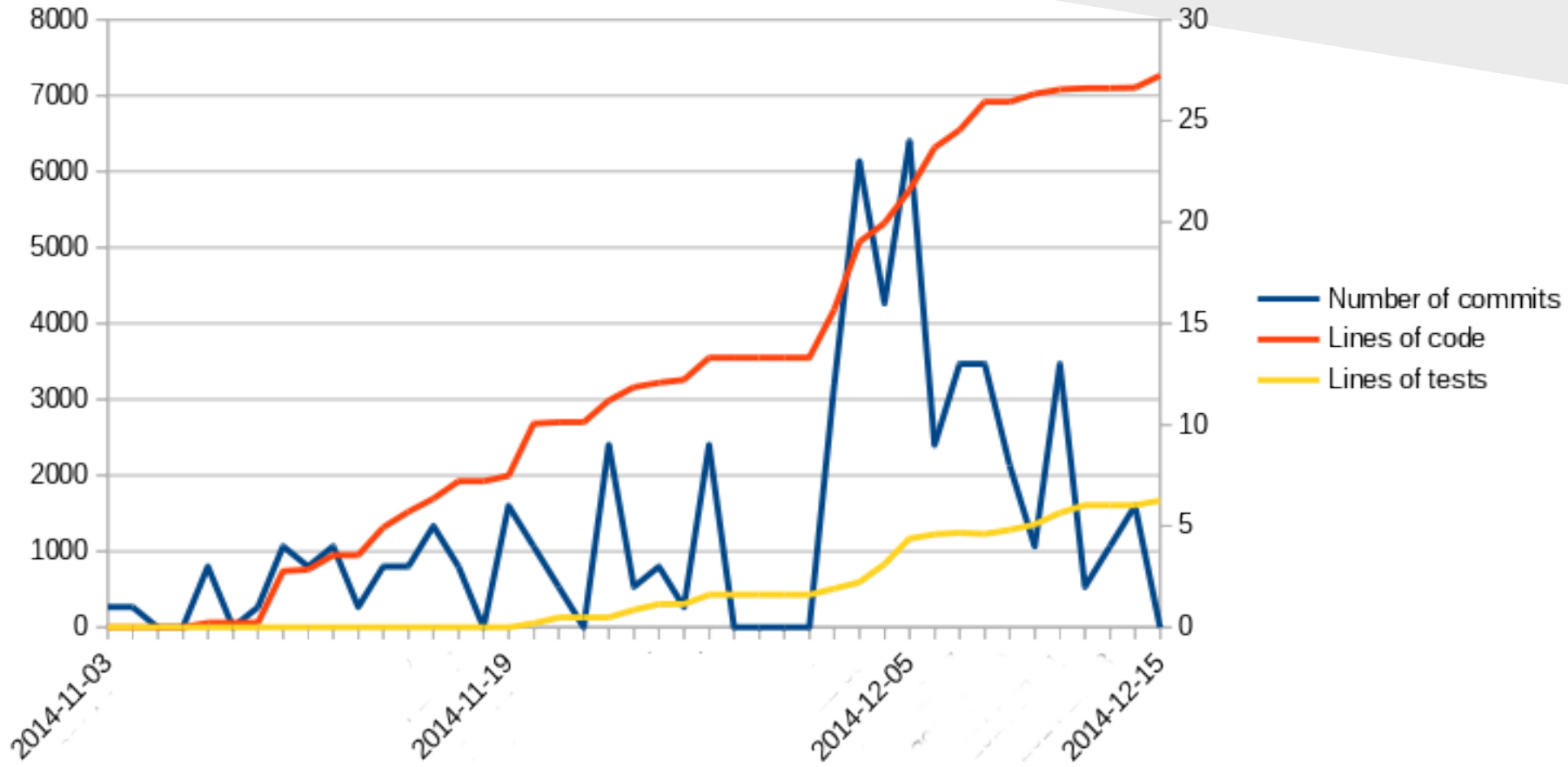
There are only 2 type of users:

- Developers (who work on project)
  - Can have the status of “Manager” of a project
- Stakeholders (who create a project)

Development roadmap

- Oct 24 - Start discussion about SRS
- Nov 4 - First “good” version of SRS
- Nov 7 - First lines of code (backend)
- Nov 13 - Committed first version of REST APIs design document
- Nov 20 - First tests on code (with test suite)
- Nov 22 - Madrid starts committing strange things
- Dec 5 - Madrid finally produces the first working page
- Dec 6 - Almost all backend functionalities work correctly
- Dec 12 - Almost all functionalities work fine in backend and frontend
- Dec 15 - Last commit

Milan commits statistics



lines of code (*.e): 7263
lines of code (*.py): 1668
commits: 215

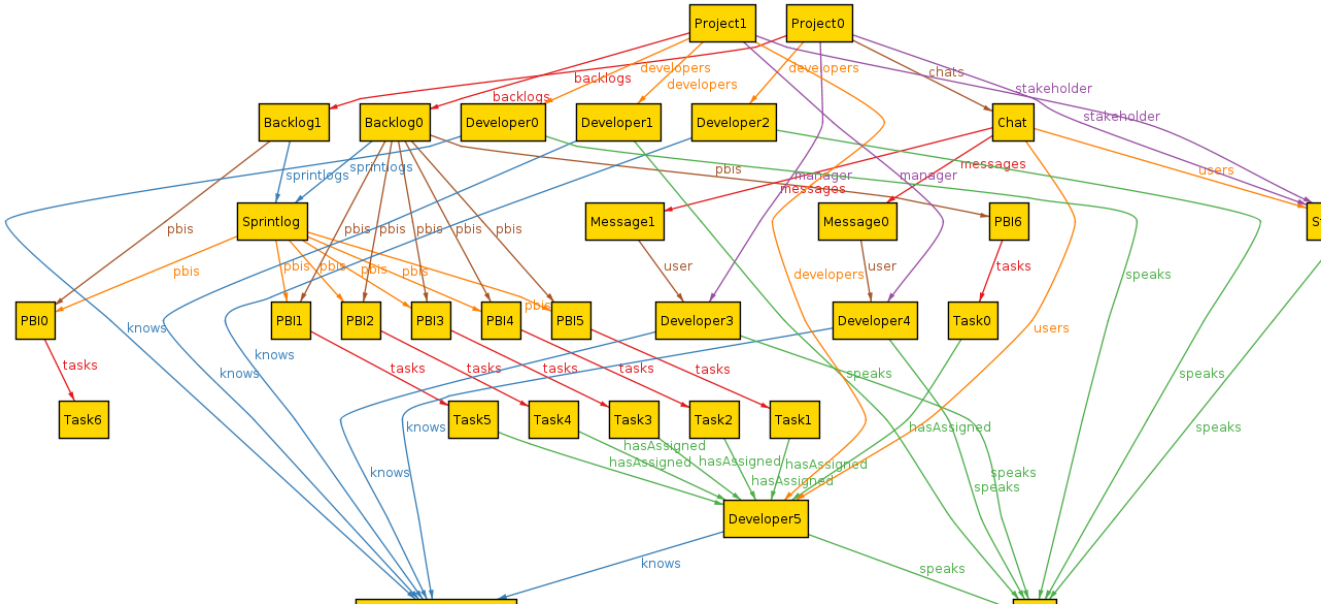
Documentation

Documents produced:

- SRS document (by Kazan)
- Alloy script and representation
- Entity-Relationship for db
- Physical project for db
- UML class diagram
- REST APIs reference
- Code reference (exported via Eiffel tools)
- The final report

Documentation - Alloy

```
sig Lang {}
sig ProgrammingLanguage {}

abstract sig User {
   speaks : some Lang
}

sig Developer extends User {
   knows : some ProgrammingLanguage
}
sig Stakeholder extends User {}


sig Project {
   manager : one Developer,
   stakeholder : one Stakeholder,
   developers : set Developer,
   backlogs : lone Backlog,
   chats : set Chat
}

sig Backlog {
   pbis : set PBI,
   sprintlogs : set Sprintlog
}

sig PBI {
   tasks : some Task
}

sig Task {
   hasAssigned : lone Developer
```



Documentation - REST APIs

Path:	/account/register
Request:	POST
Description:	Register a new user. You need to specify if the user is “stakeholder” or “developer”. In the latter case, you need to specify also “programmingLanguages” field. You can get a list of available languages invoking a request on /accout/langs NOTE: user's type cannot be changed after registration.
Authentication:	No.[1]
Parameters:	• firstname : string • lastname : string • sex : string (“M” or “F”) • dateOfBirth : Date • country : string • timezone : string (IANA format) • email : string • password : string (plain-text) • languages : string [ ] • programmingLanguages : string [ ] (only for developers) • “type” : string (“stakeholder” or “developer”)

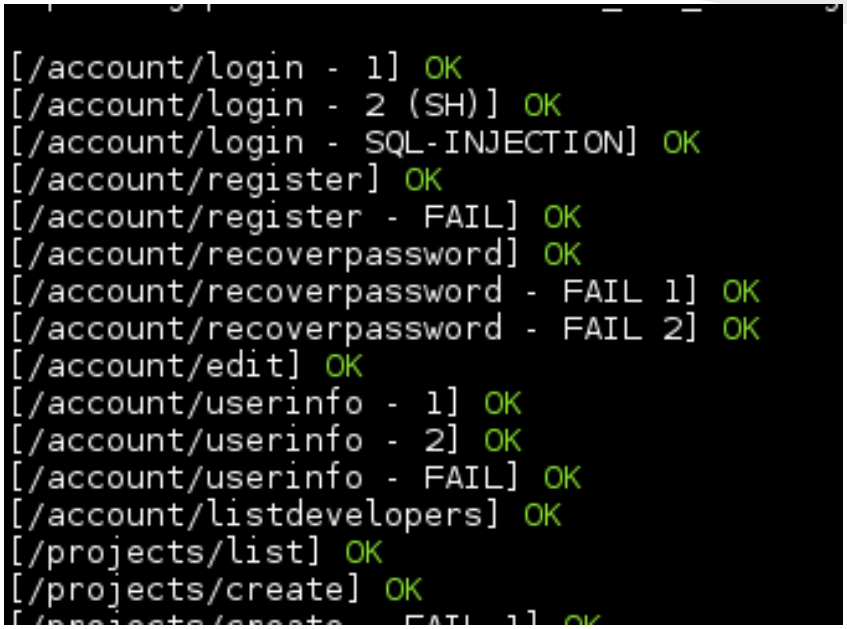
Response format:	{
	“status” : “ok”
	}

Exception 1 error code:	200 – E-Mail already exists

Exception 1 format:	{
	“status” : “error”,
	“code” : 1
	}

Testing

- We didn't use Eiffel Test Framework
  - Needed time to learn
  - Not exhaustive
- We built a simple test suite in Python
  - Simply call all REST APIs and check results
- 58 tests for a total of 1668 lines of code



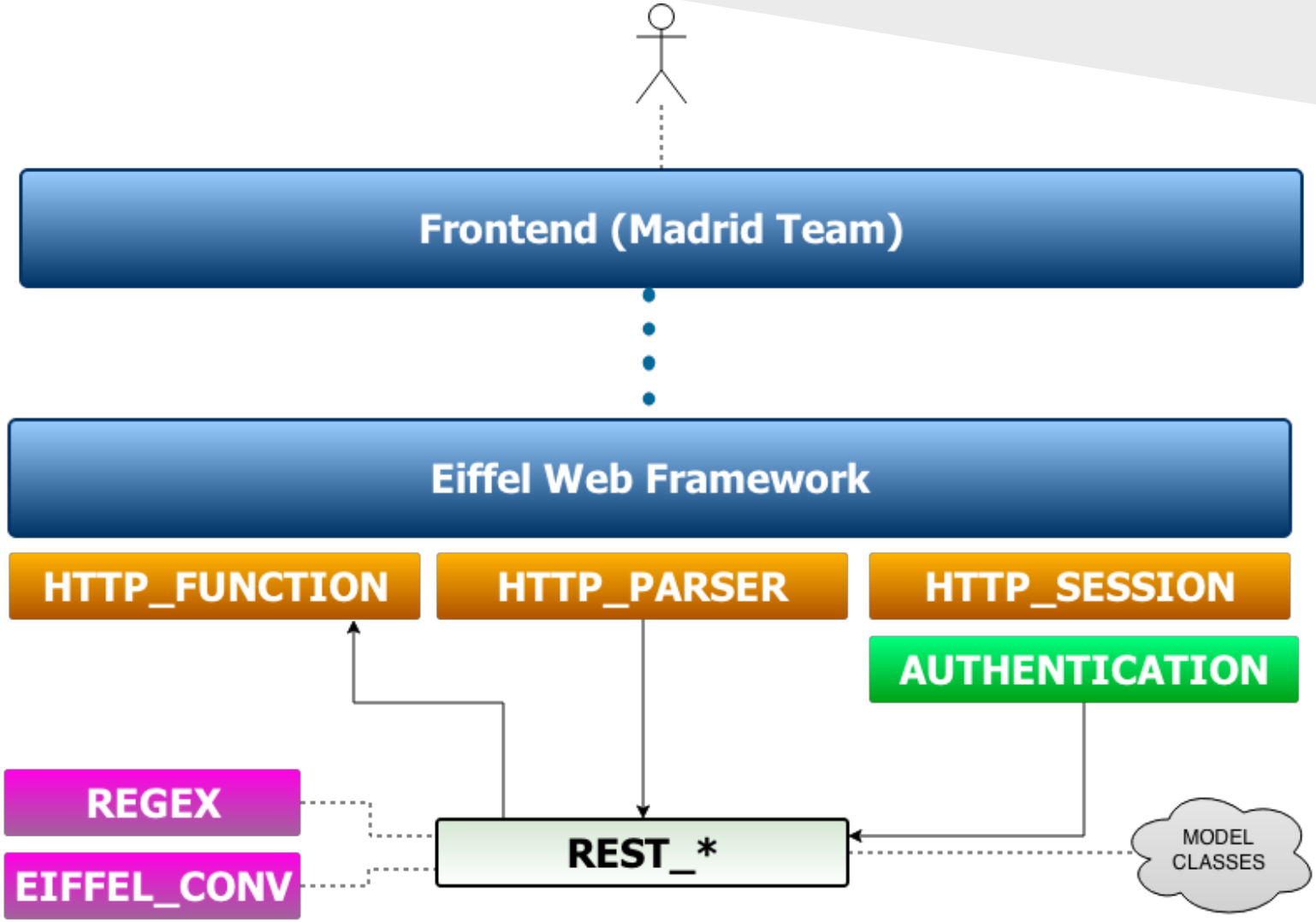
How it works

Backend Architecture

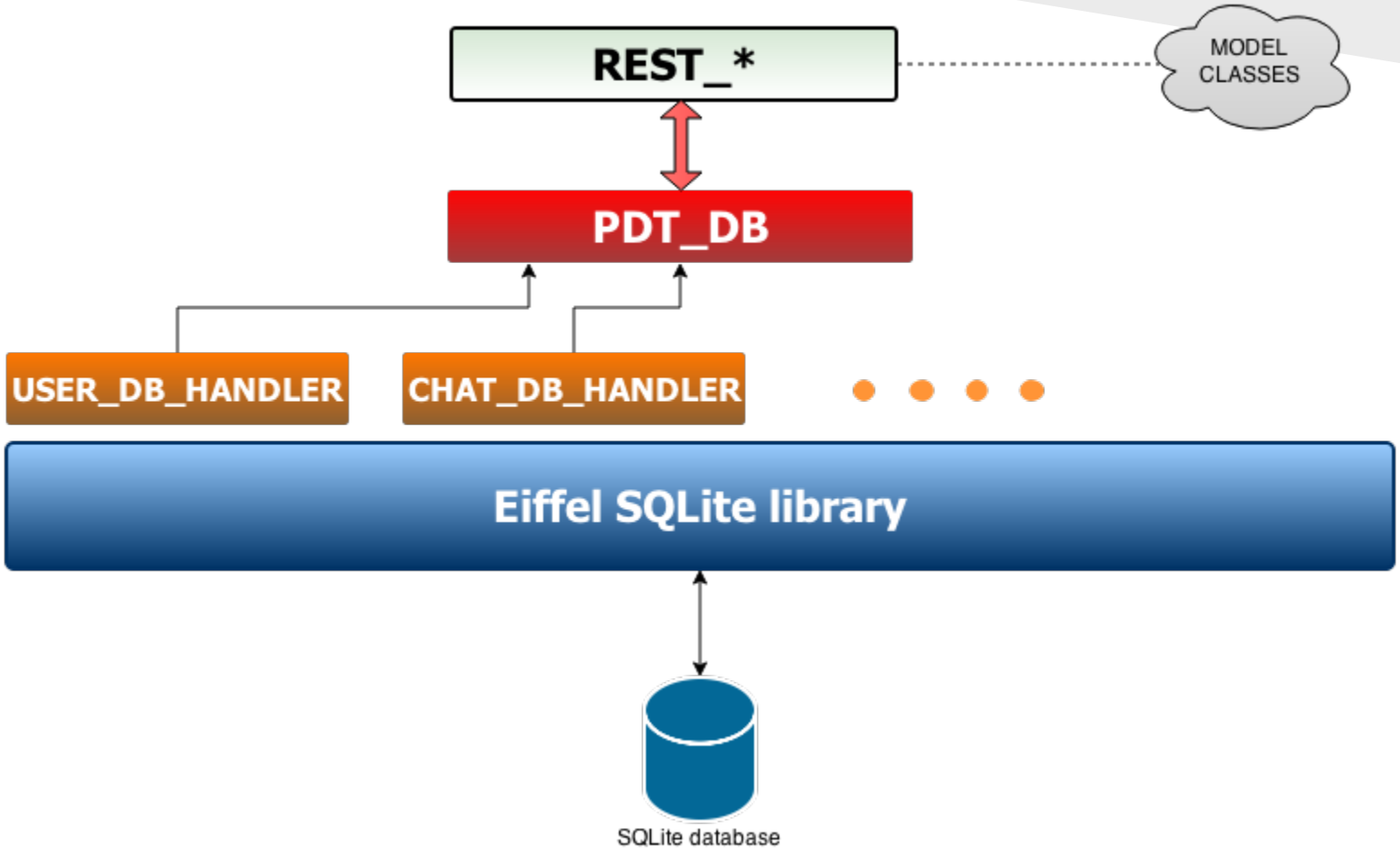
- We approached the implementation of the backend part in a bottom-up paradigm
- Levels (bottom-up):
  - Database
  - Database handlers
  - Model
  - REST APIs

Presentation of the structure is done using a top-down approach

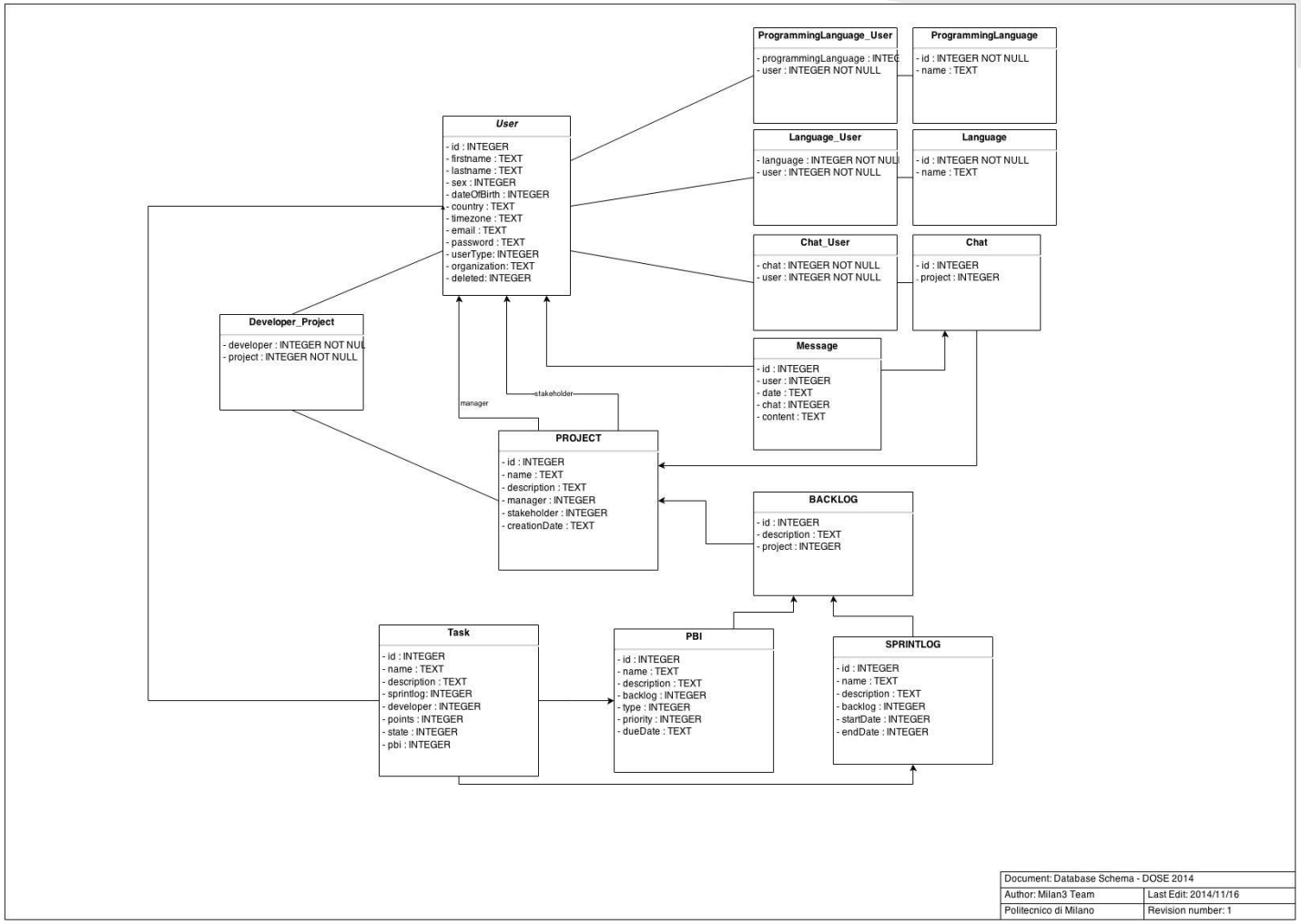
Backend Architecture



Backend Architecture



Backend Architecture



Conclusions

Considerations on Eiffel

PRO: design by contracts with instructions like require, ensure, etc.

CON: very good debugger with a lot of functions

CON: debugger is buggy and doesn't work

CON: EWF is still in development and has no documentation;
we found and reported 3 bugs in the web library

CON: Some language constructs too intricate (e.g. casting, no return)

Conclusions

- All requirements requested implemented
- Added 3 new requirements
- Good team-work and group-work (except when Madrid disappeared)
- The software works, but it needs another month of work to be at a stable and production version.