Plunner

Final Report Document

Version 1.0

Table of contents

1. Introduction

1.1 Abstract

Plunner is a new meeting planner addressed to small and medium organizations and businesses. It uses industry standard technologies like CalDAV.
The key feature of Plunner, with respect to existing ones, is the optimization of meeting times: for every meeting created, Plunner tries to find for it the right starting time by maximizing the number of its participants. Meetings' optimization and easy management increase the value and decrease the costs for organizations and businesses

1.2 Keywords

- Operations Research
- Optimization
- CalDAV
- Meeting planner
- Organisations
- Business

1.3 Document Purpose

This document is intended to illustrate the features, the concepts and the development process behind the project called "Plunner". The focus is on the ideas that inspired the project, the requirements, the design and the implementation of the project, the software engineering strategy used. Besides, we describe how it has been tested and how the experience of its development was from the point of view of the team behind it.

1.4 Background

In businesses and organizations meetings are essential for producing value and guaranteeing the quality and correctness of a product or a service. Organizing these meetings can be tedious and time consuming since different needs have to be harmonized in order to find a suitable date and time. In addition to that, the tools usually used to organize meetings are not powerful or intuitive enough for the needs of modern businesses or organizations.

1.5 Project goals

Plunner is a web application that aims to solve the problems underlined above, by providing a flexible and intuitive way to plan and organize meetings for businesses and organizations. Plunner has optimization and simplicity in mind, so that:

- Meetings can be planned by importing schedules from external calendaring services using CalDAV or by composing in-app schedules
- Meetings times and dates are determined automatically by optimizing (maximizing) the number of participants
- All the functionalities of the application can be accessed using a simple, intuitive and responsive web interface

1.6 Customer and Target

The customer of the project is Mr. Michael Young, Associate Professor at the University of Oregon, United States. The project's team has kept contact with him using emails in order to get suggestions about the project and also to perform a validation process.
Plunner's targets consist in small and medium businesses and organization with the chance, in the future, to expand to large businesses and corporations. This choice impacts on the design and the functionalities of the whole application.

1.7 Resources

The team behind the project created a real domain associated with a web-server that implements and bootstraps Plunner.
For using the application, installing it or contribute to it refer to http://plunner.com/. For the repositories associated with the project refer to https://github.com/dsd-meetme.

1.8 Team

The team is distributed across different countries and universities and it is composed by:
Claudio Cardinale, Politecnico di Milano University, claudio.cardinale@mail.polimi.it
Denis Kuryshov, Politecnico di Milano University, jaibird2493@gmail.com
Jean Barré, Politecnico di Milano University, jeanperrin.barre@mail.polimi.it
Giorgio Pea, Politecnico di Milano University, giorgio.pea@mail.polimi.it
Mihovil Vinković, Mälardalen Högskola University, mihovilvinkovic@gmail.com
Emil Silađi, Mälardalen Högskola University, emil.siladi@gmail.com

1.9 Definitions, Acronyms, Terms

Keyword	Definition
Registered Organization	An organization that has registered to Plunner
Member	A member of a registered organization. This member has been registered into Plunner by its organization
Planner of a team T	A particular Member that can manage the planning of meetings for team T(one planner per team)
Team	A set of Members of a registered organization. This set can have no counterpart in the real life, it's just a schema Plunner imposes to simplify things
Schedule of a Member	A set of time slots relative to a given period(a month for example) in which the Member is busy
Sunday Midnight	Sunday at 00:00 that is to say one minute after the 23:59 of Saturday
The Service, the application	Another way to refer to Plunner

Acronym or abbreviation	Definition
GLPK	GNU Linear Programming Kit
LDAP	Lightweight Directory Access Protocol
AWS	Amazon Web Services
JWT	JSON Web Tokens
LTS	Long term support
HTTPS	HTTP over SSL
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
CalDAV	Calendaring Extensions to WebDAV, an Internet standard allowing a client to access scheduling information on a remote server.
WebDAV	Web Distributed Authoring and Versioning, an extension of the Hypertext Transfer Protocol (HTTP) that allows clients to perform remote Web content authoring operations
MVC	Model, View, Controller

2.0 Support documentation

- Plunner's Project Plan Document
- Plunner's Requirements Definiont Document
- Plunner's Design Description Document
- Plunner's Acceptance Test Plan Document
- Plunner's Test Report Document

2 Requirements

2.1 Actors

The main actors that can be identified when describing the functionalities of the application are:

- Organization
- Registered organization
- Member
- Planner of a team T

(see “Definitions, Acronyms and Terms” section for more information)

2.2 Functional Requirements

In this section the functional requirements of Plunner are defined and associated with a priority calculated using a mathematical formula.

2.2.1 Score

For each requirement is assigned:

- An effort in terms of hours
- A ROI with a fixed scale [0-100]
- The score calculated as: ROI^2/effort

For each user story the sum of the score of its requirements (TOTSCORE) is computed and the following priorities are assigned as a consequence

- 1000 <= TOTSCORE → High priority(HIGH)
- 500 <= TOTSCORE < 1000 → Medium priority(MED)
- TOTSCORE < 500 → Low priority(LOW)

If a requirement has a ROI >= 90 then it is considered required.

2.2.2 Requirements

ID	STORY			TOTSCORE
1	**As an organization I want to register and login to the service**			**5950 HIGH**
REQ ID	**REQUIREMENT DESCRIPTION**	**EFFORT[h]**	**ROI [0-100]**	**SCORE**
1	The Service must allow an organization to signup via email and password	5	100	2000
2	The Service must allow a registered organization to login via email and password	5	100	2000

3	The Service must allow a registered organization to recover its login credentials	5	80	1280
~~4~~	~~The Service must allow a registered organization to use a two factor authentication~~	~~10~~	~~30~~	~~90~~
5	The Service must allow a remember me functionality	10	20	40
~~6~~	~~The Service must assign to a just registered organization a service specific domain~~	~~15~~	~~90~~	~~540~~
~~7~~	~~The service must verify company email~~	~~5~~	~~35~~	~~245~~
8	The service must allow to see and edit personal information	5	70	980
9	The service must allow to change the password	5	70	980

ID	**STORY**			**TOTSCORE**
2	**As a registered organization I want to register/import and to manage my members**			**1722.5 HIGH**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT[h]	ROI [0-100]	SCORE
1	The Service must allow a registered organization to register to the service, within the organization's context, its members with an email address and a password	25	100	400
~~2~~	~~The Service must allow a registered organization to import to the service, within the organization's, its members using a standard protocol(LDAP)~~	~~35~~	~~70~~	~~140~~
3	The Service must allow a registered organization to add/remove a Member and to manage its credentials	40	90	202.5
4	~~The Service must allow a registered organization to automatically send via mail its service specific domain to every new registered member. This email will contain also the password the organization chose for the member~~	~~5~~	~~70~~	~~980~~

ID	**STORY**			**TOTSCORE**
3	**As a registered organization I want to create and manage Teams**			**2516.666 HIGH**
REQ ID	**REQUIREMENT DESCRIPTION**	**EFFORT[h]**	**ROI [0-100]**	**SCORE**
1	The Service must allow a registered organization to organize its members in teams	20	90	405
2	The Service must allow a registered organization to choose for each team one of its member to become a planner	10	85	722.5
3	The Service must allow a registered organization to change the planner of each team	15	80	426.6666
4	The Service must allow a registered organization to change the name and the composition of each team	20	80	320
5	The Service must allow a registered organization to delete teams	10	75	562.5
~~6~~	~~The Service must allow a registered organization to merge teams or split teams in subteams~~	~~20~~	~~40~~	~~80~~

ID	**STORY**			**TOTSCORE**
4	**As a Member I want to login to the service within the context of my organization**			**1223.3333 HIGH**
REQ ID	**REQUIREMENT DESCRIPTION**	**EFFORT[h]**	**ROI [0-100]**	**SCORE**
1	The Service must allow Members to login within the context of their organization using the mail and the password their organization used to register them	30	100	333.33333
2	The Service must allow Members to recover their login credentials	10	70	490
3	The Service must allow the Remember me functionality for Members	10	40	160

4	~~The Service must allow Members to use a two factor authentication~~	~~15~~	~~60~~	~~240~~
5	The Service must allow a Member to see and edit personal information	5	60	720
6	The Service must allow a Member to change the login password	5	60	720

ID	**STORY**			**TOTSCORE**
5	**As a Member i want to import schedules from external services and manage them**			**1582.8333 HIGH**
REQ ID	**REQUIREMENT DESCRIPTION**	**EFFORT[h]**	**ROI [0-100]**	**SCORE**
~~1~~	~~The Service must allow Members to upload schedules as files of a standard format~~	~~20~~	~~70~~	~~245~~
2	The Service must allow Members to import schedules from external services using, the CalDAV protocol	30	90	270
3	The Service must allow Members to keep schedules imported from external services in sync with those services	30	80	213.33333
4	The Service must allow Members to keep track of the imported schedules	50	60	72
5	The Service must allow Members to remove imported schedules	10	60	360
6	The Service must allow Members to enable or disable imported schedules	10	65	422.5

ID	**STORY**			**TOTSCORE**
6	**As a Member I want to compose a schedule**			**675 MEDIUM**
REQ ID	**REQUIREMENT DESCRIPTION**	**EFFORT[h]**	**ROI [0-100]**	**SCORE**
1	The Service must allow Members to compose a schedule by specifying a name and a series of time slots of a given period in which they are busy	30	90	270
2	The Service must allow to remove, change and keep track of composed schedules	20	90	405

ID	**STORY**			**TOTSCORE**
7	**As a Member I want to export a planned meeting for my teams to my schedules**			**960 MEDIUM**
REQ ID	**REQUIREMENT DESCRIPTION**	**EFFORT[h]**	**ROI [0-100**	**SCORE**
~~1~~	~~The Service must be able to write a planned meeting to schedules imported from external calendaring services~~	~~15~~	~~60~~	~~240~~
~~2~~	~~The Service must be able to generate a file with the Member's general schedule. This schedule will contain also the planned meeting~~	~~5~~	~~60~~	~~720~~

ID	**STORY**			**TOTSCORE**
8	**As a Member I want to receive notifications about meetings planned for my teams**			**386.66666 LOW**
REQ ID	**REQUIREMENT DESCRIPTION**	**EFFORT[h]**	**ROI [0-100]**	**SCORE**
1	The Service must be able to send to Members notifications about the status of a meeting to be planned in which they're involved	15	70	326.66666
2	~~The Service must be able to send to Members notifications about the necessity of an input to help planning a meeting~~	~~15~~	~~30~~	~~60~~

ID	**STORY**			**TOTSCORE**
9	**As a planner of a team T I want to create meetings for that team**			**1719.6666 HIGH**
REQ ID	**REQUIREMENT DESCRIPTION**	**EFFORT[h]**	**ROI[0-100]**	**SCORE**
1	The Service must allow a Planner of a team T to create a meeting for that team, specifying a title, a brief description, a duration and a set of possible meeting times	30	100	333.33333
~~2~~	~~The Service must allow a Planner of a team to to create a meeting for that team, specifying which Members of the team are required and/or invited~~	~~20~~	~~80~~	~~320~~

~~3~~	~~The Service must support the creation of meetings planned with a given frequency(for example a team has to plan a meeting every two weeks)~~	~~20~~	~~70~~	~~245~~
4	The Service must support meetings to be planned taking in consideration the different time zones of the Members	10	80	640
5	The Service must decide a starting time for a meeting by maximizing the number of participants, taking also in consideration the duration of the meeting	50	80	128
~~6~~	~~The Service must support meetings to be planned with given constraints (for example, the number of invited Members has to be > k)~~	~~30~~	~~40~~	~~53.333333~~

ID	**STORY**			**TOTSCORE**
10	**As a planner I want to manage meetings that I created**			**1645 HIGH**
REQ ID	**REQUIREMENT DESCRIPTION**	**EFFORT[h]**	**ROI [0-100]**	**SCORE**
1	The Service must allow planners to cancel a meeting they created, this applies both to the case of planned or to be planned meetings	10	80	640
2	The Service must allow planners to change the title and the description of a created meeting; this applies both to the case of planned meetings or to be planned meetings	20	80	320
3	The Service must allow planners to change the possible times and the duration of a created meeting; this applies only to meetings to be planned	10	80	640
~~4~~	~~The Service must allow planners to change the constraints of a created meeting~~	~~20~~	~~30~~	~~45~~

2.3 Non functional Requirements

2.3.1 Availability

Considering that:

1. Businesses and organizations can have offices and employees in different part of the world
2. Companies may need to plan a(virtual) meeting that involves people from different offices in different locations with different timezones

Plunner must be available 24 hours a day, 7 days a week

2.3.2 Security and privacy

Plunner must manipulate and retrieve from external services only the data that is necessary to accomplish the planning of meetings so that the privacy of the registered organizations and their members is guaranteed.
The data Plunner exchanges via web with Members and Registered Organizations may be manipulated and used in a fraudulent way, to deny this scenario **Plunner must use secure communication protocols.**

2.3.3 Uptime and data redundancy

Since meetings are crucial instruments for businesses and organizations to produce value, Plunner should guarantee high availability and data redundancy. Since this application will be created in the context of the DSD course, the team will not be able neither to build nor to use a dedicated infrastructure for it and so estimating and prooving exact values for data redundancy and uptime is not possible; however, in the case that there's the chance to build and test a dedicated infrastructure an uptime of at least 99.99% is desirable along with at least one database replication.

2.3.4 Performances

Since the user base of the application is potentially high, Plunner must be able to manage an high volume of requests.
Since this application will be created in the context of the DSD course, the team will not be able neither to build nor to use a dedicated infrastructure for it and so estimating and prooving exact values for performances is not possible; however, in the case that there's the chance to build and test a dedicated infrastructure, is desirable that Plunner can manage at least 1000 requests/h.

3 The development process

3.1 SCRUM overview

The team has adopted the SCRUM software development methodology. SCRUM addresses the problem of requirements volatility and unpredicted changes in a software project through a series of tools, ideas and organization rules.
This methodology has been chosen for the following reasons:

- It organizes the development process in a way that the problem of requirements volatility and unpredicted changes to the software project is addressed in a robust and relatively simple manner
- It promotes team working and communication and so it's great to know new people from different cultures and countries
- It imposes a systematic way to approach to the different steps of the development process through sprints, meetings and sprint backlogs
- It is document oriented which is useful for the future careers of the members of the project's team and for the Score contest

3.1 SCRUM implementation and project plan

3.1.2 Project Plan

14/11/2015 to 28/11/2015: **First Sprint + Alpha Version**
28/11/2015 to 12/12/2015: **Second Sprint + Beta Version**
12/12/2015 to 26/12/2015: **Third Sprint**
26/12/2016 to 7/01/2016: **Refinements, documentation upgrade**
7/01/2016: **Final Version**

3.1.3 Scrum roles

- **Product owner:** Claudio Cardinale
- **Scrum master:** Giorgio Pea
- **Development Team:** All members of the project team

3.1.4 Development Team organization

Backend team:
Claudio Cardinale, controller/model developer, architecture manager
Emil Siladi, model developer
Mihovil Vinkovic, model developer
Denis Kuryshov, controller developer

Frontend team:
Giorgio Pea, view and logic developer
Jean Barre, view developer

3.1.5 Contact with the customer

The customer has been contacted only two times: the first time to discuss a target change for the application (the target was changed towards small organizations and businesses) and the second time to have an opinion about the final product. The communication has been carried out via email. This frequency of communication has been caused by the low time available for the team and by the fact that the description of the project provided by the Score was clear and understandable.

3.1.6 Daily meetings, sprint review meetings

The team has decided that during each sprint the daily scrum meeting will be held everyday at 18:30 CET/CEST (according to european DST).
The team has decided that at the end of each sprint a sprint review meeting will be held in the afternoon in a convenient time for each team member.

3.1.7 Activities performed before the starting of the first sprint

Before the starting of the first sprint, the project team has completed the following tasks:

- Requirements gathering and discussion
- Choice of communication tools, meetings frequency, sprints duration
- Choice of development roles
- Choice of SCRUM roles
- Choice of programming languages and technologies
- Choice of a suitable software architecture

3.1.8 Sprints

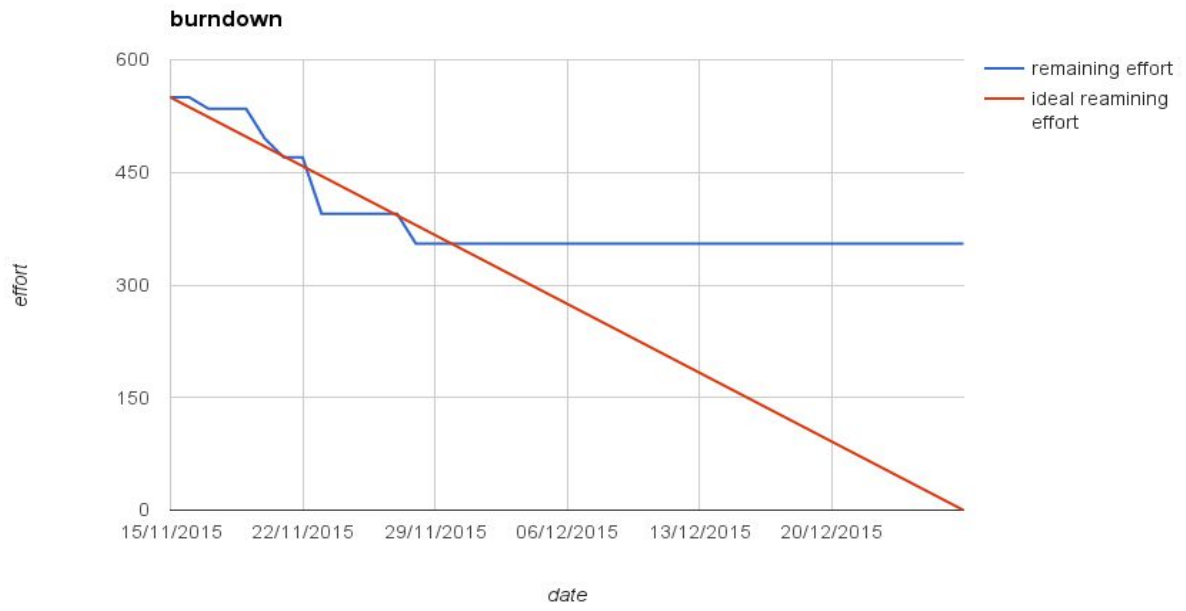
The project team has decided to make 3 sprints of a 2 weeks length, this length has been chosen because it guarantees a good balance between the necessity of being flexible and adapts to changes and the necessity of a sufficient quantity of time for development with quality and robustness the product. The tasks assigned to each sprint have been chosen taking in consideration the score associated to each requirements of Plunner, as detailed in the “Requirements” section of this document.

Sprint 1 (from 14/11 to 28/11)

Requirements to be completed	Was it completed?
Organization Login and Registration (Reqs 1.1, 1.2,1.3)	YES
Organization’s members registration, management and login (Reqs 2.1, 2.3, 4.1, 4.2, 4.3)	YES
Organization’s teams creation and management (Reqs 3.1, 3.2, 3.3, 3.4, 3.5)	YES
An organization’s member can import his/her schedules from an external	NO

calendaring service using the CalDav protocol (Reqs 9.1, 9.2, 9.3, 9.4, 9.5, 9.6)	
A planner can plan a meeting for its associated teams (Reqs 10.1, 10.2, 10.3)	NO

Period	**Hours invested**
First Sprint - first week	155
First Sprint - second week	167
	TOT: 322



Comments to this sprint:

Some requirements have not been completed in time and have been delayed to the second sprint. After this first sprint the development team has realised that is necessary to increase the quantity of work and to be more synergetic.

Extraordinary event happened during this sprint:

An original team member left the project (Joan Josep Crespi Villalonga) because he was not be able to sustain the required work rhythm.

.

Sprint 2 (from 28/11 to 12/12)

The following requirements have been added to the project: 1.8, 1.9, 4.5, 4.6

The following requirements have been dropped: Reqs 1.4, 1.6, 1.7, 2.2, 2.4, 3.6, 4.4, 5.1, 7.1, 7.2, 8.2, 9.2, 9.3, 9.4, 9.6, 10.4

The dropping of the above mentioned requirements has been decided for reasons of time and because of the common will of focusing on the meetings' starting time optimization as the main feature of the product.
At the end of this sprint all the requirements of the project have been completed.

Period	**Hours invested**
Second Sprint - first week	196,5
Second Sprint - second week	224,5
	TOT: 421

Comments to this sprint:

The team increased the work rhythm but some lack of communication emerged, nevertheless it has managed to complete all the requirements and also update some documentation.

Sprint 3 (from 12/12 to 26/12)

This sprint has been dedicated to testing, polishing and verification of the product. During this sprint the frontend team has completed the design of the web interface of Plunner in a way that it adapts to different screen resolutions (responsive design).

Period	**Hours invested**
Third Sprint - first week	81,5

Third Sprint - second week	91
	TOT: 172,5

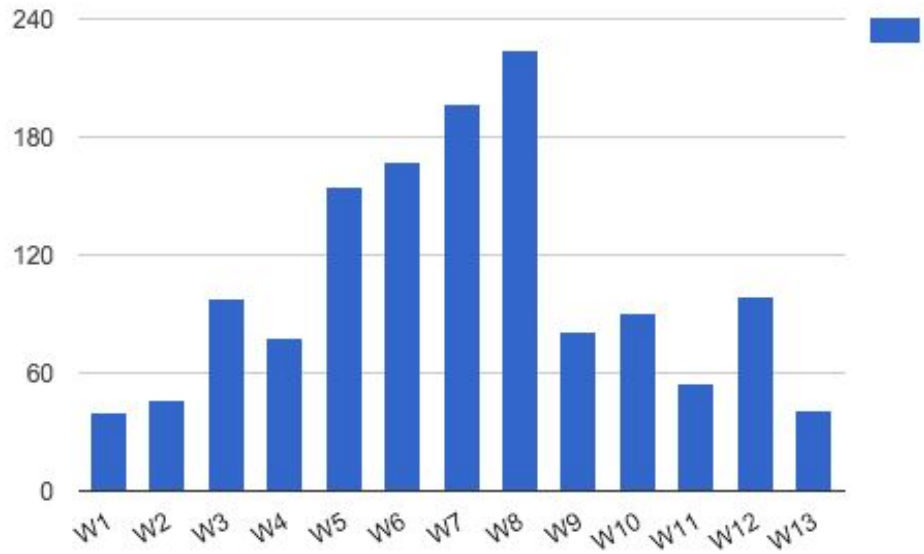
Comments to this sprint:

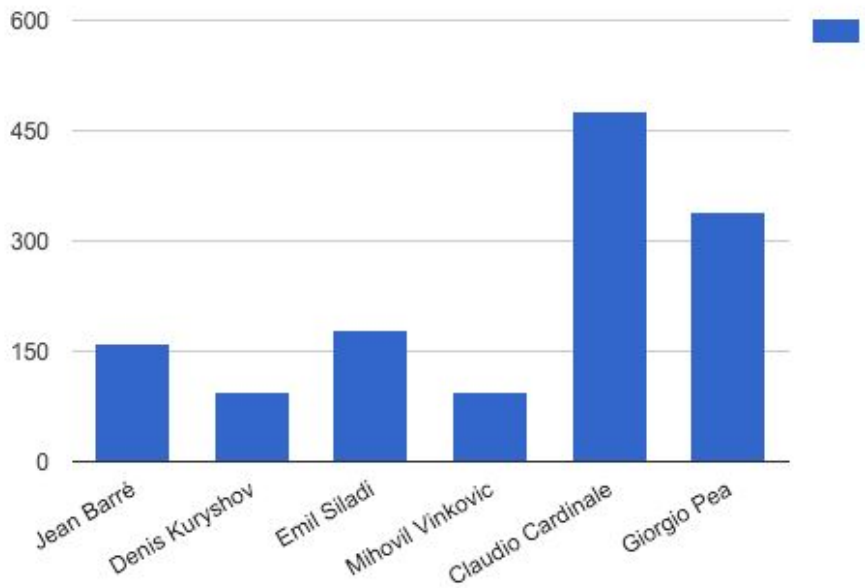
The lack of communication seemed not fixed and some members of the team have not contributed in a significant way to the project during this period, nevertheless the final product resulted after the sprint looked like valid, complete and interesting.

3.2 Process statistics

3.2.1 Working hours

TOT: 1373h





3.2.2 Contributions

Backend team:

Claudio Cardinale: 433 commits / 17,336 ++ / 7,844 --
Emil Siladi: 86 commits / 2,186 ++ / 1,060 --
Mihovil Vinkovic: 33 commits / 1,641 ++ / 1,226 --

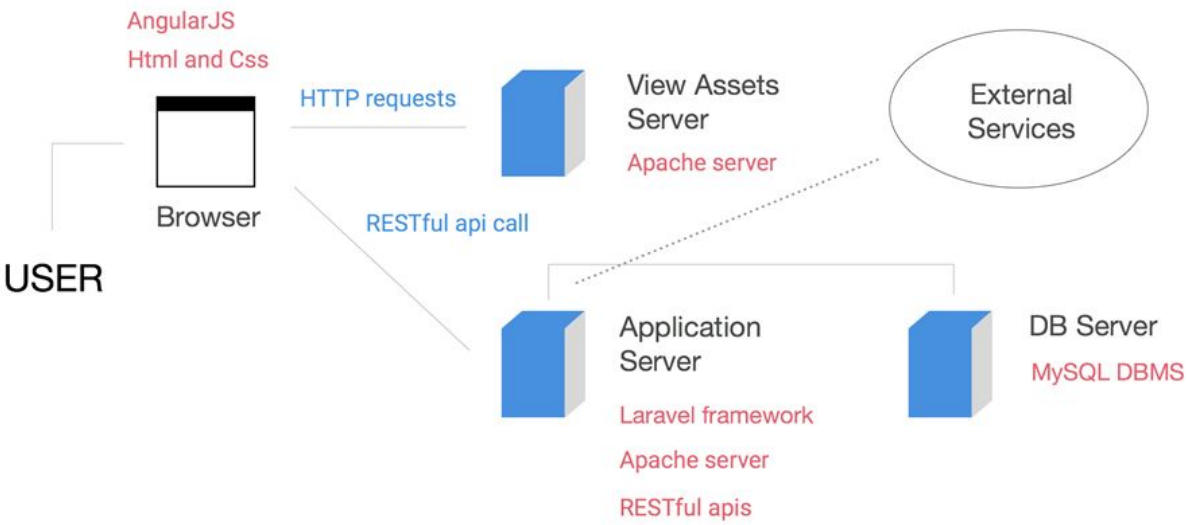
Frontend team:

Giorgio Pea: 309 commits / 112,663 ++ / 105,155 --
Jean Barré: 43 commits / 9,470 ++ / 6,103 --

sources:
https://github.com/dsd-meetme/backend/graphs/contributors,
https://github.com/dsd-meetme/frontend/graphs/contributors

4 Design and implementation

4.1 Architecture



As visible from the picture above, Plunner adopts a 3 tier* client server architecture, the main reasons behind this choice are the following ones:

- **Scalability:** the nodes composing the application server and db server can be increased in number without the need of redesigning the whole system
- **Reliability:** since the chosen architecture is distributed, failures can be easily isolated and database replications can be implemented without impacting too much on the design of the whole system
- **Adaptability:** the chosen architecture can be easily enriched with a load balancer that distributes and activates nodes in relation to the number of incoming requests.
- **Security:** since the chosen architecture strictly divides business logic and data and since the application server and the db server can be easily isolated from the web using firewalls, malicious attacks can be prevented and high level of security granted

**The web assets server is not considered in the account of tiers since its only purpose is to deliver via the web the code that after being executed by a browser implements the frontend part of Plunner*

4.1.1 Tiers

- **Application Server:** a tier that manages and implements the business logic of the entire application, it process and answers to requests from client's browsers
- **DB Server:** a tier that manages the persistent data of the application
- **Browser:** a tier that represents the layer of interaction between the client and the application

4.1.2 MVC

Both for the frontend and backend the team has decided to adopt a design pattern known as MVC (Model View Controller). This choice has been made since the pattern is one of the most used and tested in the industry and it lets the team split the work better and in a more systematic way.

4.1.3 Client Server Communication

The communication between the client and server is implemented via a remote set APIs that the server makes available to the client. This communication method has been chosen for reasons of flexibility, security and because it is easier to implement.

4.1.4 External services

Since Plunner is a calendar based application, an integration with external calendaring services has been designed; the purpose of such integration is to give to the client the chance to easily import schedules and facilitate the meeting planning. The team has thought about using a standard and well diffused/supported protocol for integrating external calendaring services and so during the implementation phase this has been taken in consideration.

4.1.5 User interface

The team has decided to adapt a sleek, modern, minimalistic and business oriented web interface that must be responsive.

4.2 Technologies

4.2.1 Browser tier

HTML, CSS, Javascript, AngularJS framework

4.2.2 DB server

MySQL, AWS (as future improvement for High traffic)

4.2.3 Application Server

PHP, Laravel framework (LTS version), Apache Server, AWS (as future improvement for High traffic).

4.2.4 Client Communication

Remote set of apis implemented via the Laravel Framework following the RESTful standard.

4.2.5 External Services

External calendaring services integration is made using the CalDAV protocol.
The application respects user's privacy and only reads the timeslots (meaning the starting and ending times of every timeslot) in the calendars.
These technologies were chosen by the team for the following reasons:

- They are easy to learn and use
- They are powerful and well diffused (standard open protocol)
- Some members of the team had previous experience with them

4.3 Optimization

The team has decided, exploiting the knowledge in operations research of some of its member, to include in the project and optimization process consisting in the selection of meetings' starting times by maximizing the number of participants.
By analyzing the computation of a starting meeting time according to each participant's schedules, we defined our problem through an linear program. We used GLPK, an open source program to resolve the optimization problem, maximising the number of people to attend the meeting. Below the mathematical model for implementation is detailed.
We maximise the number of people attending a meeting. We define two binary vars:
xij = {1 : person i attends meeting j; 0 : other cases}
yij = {1 : meeting i is planned to start at timeslot j; 0 : other cases}
In fact since we had to transform timeslots in a discrete form we decided to consider a timeslot like an entity with a fixed time, for example we can have the following time zones (every 15 minutes):
1 -> 01/12/15 00:00; 2 -> 01/12/15 00:1; 3 -> 01/12/15 00:30 ...
We have also the following parameters:

- UserAvailability: shows the timeslots when the user is available
- MeetingsAvailability: shows the timeslots when the meetings can be planned

- UsersMeetings: shows which users should participate to the meeting

We also have a list:

- MeetingsDuration: says the duration of a meeting

So we optimize: $max \sum_{i \in Users; j \in Meetings} (x[i, j] \cdot MeetingsDuration[j])$

Here we have the complete model (with all constraints):
https://github.com/dsd-meetme/backend/blob/master/app/Console/Commands/Optimise/model.stub

With this optimization we are able to plan all meetings automatically, but to do that we have to add some constraints (these are not the linear programming constraints) to work in a real environment.

- A planner can plan a meeting for the weeks after, but a meeting must be planned inside one week
- The optimization task is done on Sunday at midnight for the next week then automatic email notifications are sent
- The week starts on Monday
- A planner can plan a meeting for the next week until Sunday midnight, since the optimization is done on Sunday at midnight for the next week
- After the optimization is not possible to add new meetings to that week, but it is possible to remove and update (information not date)
- A user can update the busy timeslots for the week after until Sunday midnight, like planning. He can always compile busy timeslots for the weeks after the next one
- Timeslot is multiple of 15 minutes
- We always use UTC time

We have to proceed in this way since we decided to perform the optimization task one time each week and it optimises the week after.
Counterexamples:

- if we select a longer period, meetings with availability of the next week are often planned for it (probably in the next week they are able to find more employees), but at the same time on the week after we have some meetings that are already planned and we cannot optimise them again, so we cannot optimise the week properly because we have old meetings planned that are like busy timeslots. If we don't plan them we are able to optimise the entire week with them (shifting them)
- if we select a lower period we don't have enough timeslots to perform a good optimization in fact for example if we choose a period of 3 days we have few timeslots available to shift meetings and we don't have the best optimization

Improvements:

- Customization: the company can choose the frequency of optimization task
- At the moment we can have meetings with only one employee (useless)
- Different kind of users (required and invited) given priority to one type
- Concentrate meetings to avoid empty timeslots
- If we have optimization errors -> rollback for that company

- Concentrate meetings to avoid empty timeslots

5. Future improvements

5.1 Other external calendar services support

Members would appreciate a possibility to link their accounts with non CalDAV supported calendaring application like Google Calendars or Microsoft Exchange and this can be done via specific APIs.

5.2 LDAP or Lightweight Directory Access Protocol

This data exchange protocol is widely diffused among businesses and corporation, it would be a nice and business oriented addition to the project's features

5.3 HTTPS

The security the application offers can be greatly increased using HTTPs, this technology was not adopted for economical reasons

5.4 Future Plunner architecture with mobile application

Plunner's architecture can be extended with a mobile application for all the major mobile operating systems(iOs, Android, Windows Phone).
Since during the development the team has used RESTful api, the extension described above seems simple.

5.5 Improve testing and code quality

The project's backend and frontend unit tests coverage and quality can be increased, automated end to end tests built and more stress tests performed

5.6 Unimplemented requirements

The requirements dropped during the second sprint of the project can be implemented

5.7 Meetings prioritization

Meetings can be prioritized in order to extends and facilitate the meetings' times optimization process

6. Verification and validation

The verification and validation of Plunner has been conducted by the development team using automatic tests and manual test with the support of different automated tools.
Every backend and frontend team member has worked in its own branch. After successful testing, the written code is pushed into the master branch.

6.1 Backend Tests

The tests performed by the backend developers on the backend of Plunner consist in:

- Unit tests for the different modules of the backend of the application. These tests should cover at least 80% of the codebase
- Manual tests for the the communication and interaction of the different modules of the backend of the application. These tests have been preferred to automated and more systematic integration tests for reasons of time and complexity
- Manual tests for the optimization module of the backend of the application

6.2 Frontend Tests

The tests performed by the frontend developers on the frontend of Plunner consist in:

- Unit tests for the different modules of the frontend of the application. These tests should not be as extensive as the ones for the backend of Plunner, this because unit testing of the frontend is very complex and time expensive
- Manual tests for the the communication and interaction of the different modules of the frontend of the application. These tests have been preferred to automated and more systematic integration tests for reasons of time and complexity
- Manual tests to check the usability of the web interface
- Manual tests to check the responsiveness of the web interface: the web interface should adapt itself to different screen resolution and guarantee by doing so an acceptable usability
- Manual tests to check the compatibility with different browsers: the web interface should be as much consistent as possible with different browsers

6.3 End to end tests

For reasons of time and complexity, end-to-end testing should be done manually by one guy from the frontend and another one from the backend to check the correct behaviour of the global system

6.4 Testing tools

6.4.1 Backend testing tools

The main tools used by the backend developers to perform tests are:

- PHP Unit (https://phpunit.de/)
- Scrutinizer (https://scrutinizer-ci.com/)
- Travis (https://travis-ci.org/)

6.4.2 Frontend testing tools

The main tools used by the backend developers to perform tests are:

- Karma (https://karma-runner.github.io/0.13/index.html)
- Jasmine (http://jasmine.github.io/)
- Travis (https://travis-ci.org/)

6.5 Acceptance test plan

A simple acceptance test plan has been made so that the application is accepted if:

- It guarantees all the requirements (manual test + unit tests)
- It has a responsive web interface (manual tests)
- Its frontend part supports different browsers (manual tests + frameworks browser support specification)

7 Distributed project experience

7.1 Things that should have been improved

- Communication among team members
- The time dedicated to the project at the beginning of the course should have been increased to be able to complete earlier the documentation and focus more on the development
- Internal organization
- Attention to the project's possible risks and problems
- Testing: more modules of the frontend part of the project should have been unit tested and end to end tests using automated tools like protractor should have been implemented

7.2 Positive aspects and qualities

- Interact with other cultures and with other development ways and models
- Use a systematic and defined way to design and develop software(SCRUM)
- Complete in time a good and functional application
- The team has used the Laravel framework version 5.1 LTS (3 years of support), so that the software remains stable and well supported
- All errors are logged
- The project's files organisation is ideal for a big project
- The project can be easily configured via config files
- People testing the entire environment are different from people that developed these functionalities to give an independent way
- A simple presentation site has been made to present the product and link to installation instructions
- All time data is converted in UTC for reasons of simplicity and readability
- To prevent malicious attacks, every 30 days a user of Plunner has to sign in again even if he or she has turned on the remember me functionality
- The frontend and the backend part of the application have been developed using two repositories, so that the designed architecture of Plunner is perfectly reflected. This organization of things accelerates and facilitates the development process
- Use of external services for high resources task, like sending emails
- As authentication technology JWT has been used: it protects against attacks like CSRF or XSS and allows to reduce the amount of resources need on the backend server.
- The backend part of the application is packeged via https://packagist.org/