BusPlanner	Version: 1.0
Final Project Report	Date: 20-01-2017

Distributed **S**oftware **D**evelopment:
BusPlanner
Final Project Report

Revision History

Date	Version	Description	Author
20-01-2017	1.0	Initial draft	Isabella Agosti

Contents

1 INTRODUCTION

1.1 Purpose of this document

The purpose of this document is to summarize the experiences related to the BusPlanner project, both in terms of produced results and project work.

1.2 Document organization

The document is organized as follows:

- Section 1, *Introduction*, gives an overview of this document describing its contents, scope etc.
- Section 2, *Background and Objectives*, describes the identified problem, goals of the project and high level requirements.
- Section 3, *Organization*, describes how the project members were organized and the tools used for development and collaboration.
- Section 4, *Requirements, Design and Implementation*, describes the project's requirements, high-level architecture and algorithm used for its implementation.
- Section 5, *Development Process*, describes how the scrum process was planned for our development activities.

1.3 Intended Audience

The intended audience of this document is composed of:

- Development team.
- Team supervisors:
  - Abhilash Thekkilakattil (MDH).
  - Elisabetta Di Nitto (POLIMI).
- Project customer: Aneta Vulgarakis Feljan (Ericsson).
- Stakeholders.
- Any developer who is interested into improving our project.

1.4 Scope

This document aims to summarize all the activities regarding the DSD project and at the same time make some reflections about the development process. It also provides some metrics that will help the developers analyze the performances and the level of organization.

1.5 Definitions and acronyms

1.5.1 Definitions

Keyword	**Definitions**
User	A person who requests buses.
Fleet Manager	They own the buses and they are the resource managers of the company.
Bus Driver	The driver of the company's buses.
User Request	Request generated by the users that choose the stops where they want to get on and off the bus.
Algorithm	A method used to enhance the scheduling process (static and dynamic).
Sprint	A repeatable work cycle which is also known as iteration.
Project Customer	The customer who requested the software product.
Acceptance Test	The process of verifying that a solution works for the user.

1.5.2 Acronyms and Abbreviations

Acronym/abbreviation	**Definitions**
UI	User Interface
GUI	Graphical User Interface
MDH	Mälardalens Högskola, Västerås, Sweden
POLIMI	Politecnico di Milano, Milan, Italy
QA	Quality Assurance
DSD	Distributed Software Development

1.6 References

- Project DSD page:
  http://www.fer.unizg.hr/rasip/dsd/projects/busplanner.
- Project application page:
  https://busplanner-f496d.firebaseapp.com.
- Project repository:
  https://github.com/AndreaColombo/BusPlanner---DSD-project-2016-2017.

2 BACKGROUND AND OBJECTIVES

2.1 Background

Johannesburg has a complex bus system which is quite different from what is used elsewhere. There exist a gap of effectiveness and perfect schedule time. Technology has not yet fully revolutionized the process of scheduling as most of the processes are done by manual work.

These are the main factors that have created a problem when users wanted to schedule their everyday trips. A lot of waiting time, missed and late buses seem to be a normal day routine. Still, finding the perfect scheduling algorithm that will ease the transportation is yet a big step to be achieved.

A lot of factors, such as the number of passengers that want to use a bus, the alternative routes, etc, have been considered in this process.

2.2 Project goal

In order to solve the above mentioned problems, we implemented an algorithm that can help in the bus planning process. It increases the process' efficiency and reduces the time needed to do a scheduling. The users waiting time should be dropped from hours to minutes.

2.3 High level description of the functionalities

The BusPlanner project is based on an algorithm that simulates user requests around the city of Johannesburg. These requests are identified by the two bus stops where the user wants to get on and off the bus. Based on this information, the algorithm is able to identify which route reaches both the user's starting and end point, and then assign the user to the bus already covering that route or assign a new bus to that route if the one already covering it is full.

In a real world users will interact with the system by sending requests for a bus, related to a specific position. They will also be able to view the buses' location in the city, thanks to the mapping service the system makes use of.

On the other hand, bus drivers can see all the user requests, along with all the related information.

Finally, fleet managers are the company's resource managers: they manage buses, drivers and routes and they have access to all the information related to the past rides.

3 ORGANIZATION

3.1 Project team

Our team working on BusPlanner is composed of 7 members located in 2 universities as follows:

1. Team members located in MDH Sweden:
    - Albi Dode
    - Huy Hoang Nguyen
    - Muhammad Ejaz Khan
    - Sharvathul Hasan Ameerjan
2. Team members located in POLIMI Italy:
    - Andrea Colombo
    - Isabella Agosti
    - Stefano Antonino Badalucco

More information on the members can be found at:
http://www.fer.unizg.hr/rasip/dsd/projects/busplanner

3.2 Team roles

Team member	**Role until 01/12/2016**	**Role from 01/12/2016**
Isabella Agosti	Product Owner	Product Owner
Stefano Antonino Badalucco	Developer	Scrum Master
Andrea Colombo	Developer	Developer
Albi Dode	Scrum Master	Developer
Huy Hoang Nguyen	Developer	Developer
Muhammad Ejaz Khan	Developer	Developer
Sharvathul Hasan Ameerjan	Developer	Developer

3.3 Work division

Front-End team

- Isabella Agosti.
- Stefano Antonino Badalucco.

Back-End team

- Andrea Colombo as static algorithm developer.
- Albi Dode as dynamic algorithm developer.
- Huy Hoang Nguyen as static algorithm developer and testing.
- Muhammad Ejaz Khan for integration.
- Sharvathul Hasan Ameerjan as dynamic algorithm developer.

3.4 Customer

Our customer is Aneta Vulgarakis Feljan from Ericsson.

3.5 Supervisors

The team had two supervisors for the project:

- Abhilash Thekkilakattil - From MDH.
- Elisabetta Di Nitto - From Polimi.

3.6 Used tools

- **Skype**
  For meetings with supervisors and internal meetings.
  https://www.skype.com

- **GitHub** and **GitHub Desktop**
  To collaborate with the team and keep track of the changes in the documents.
  https://github.com and https://desktop.github.com

- **Balsamiq**
  To design the mockups.
  https://balsamiq.com

- **TeXstudio**
  LaTeX editor we used to write this document.
  http://www.texstudio.org

- **BasicTeX**
  Distribution of the LaTeX system.
  http://www.tug.org/mactex/morepackages.html

- **Google Drive**
  For sharing documents.
  https://drive.google.com

- **WhatsApp**
  For instant messages.
  https://www.whatsapp.com

- **Telerik Test Studio**
  For automatic testing.
  http://www.telerik.com/teststudio

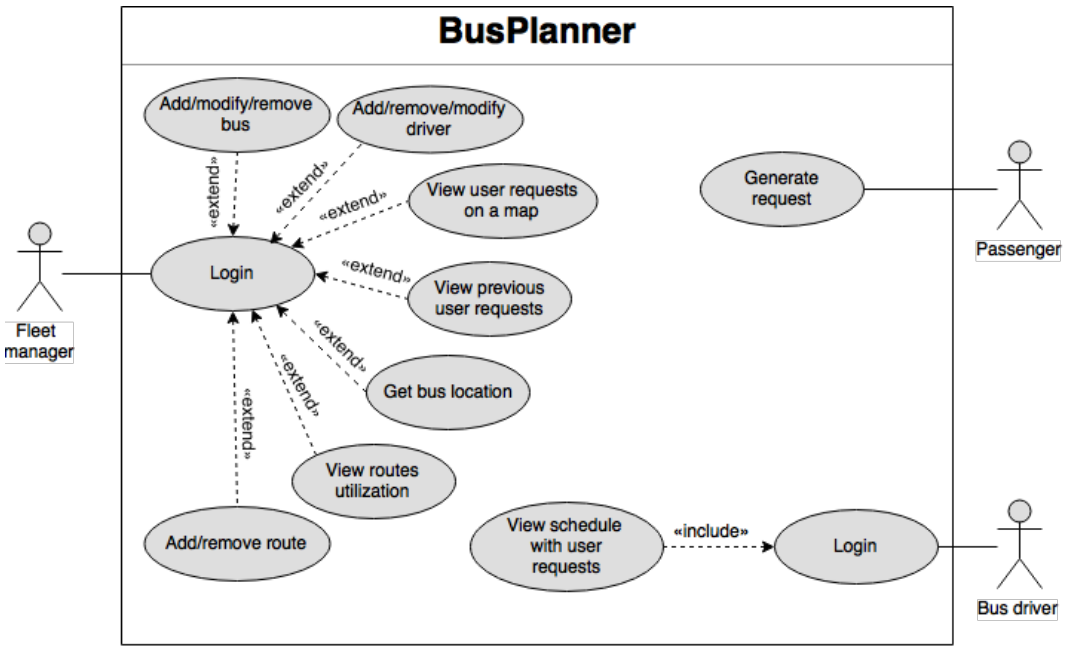
4 REQUIREMENTS, DESIGN AND IMPLEMENTATION

4.1 Actors

- **Fleet manager**, who performs the following activities:
  - Login.
  - Get bus location.
  - Add/Remove/Modify bus.
  - Assign drivers to buses.
  - Add/Remove route.
  - Add/Remove/Modify driver.
  - View the user requests on a map.
  - View previous user requests.
  - View routes utilization.
- **Bus driver**, who performs the following activities:
  - Login.
  - View schedule with user requests.
- **Passenger**, who generates the user requests for a bus, specifying at which stop he/she wants to get on and off the bus.

4.2 Functional Requirements

4.2.1 Use case



4.2.2 User stories

ID	User story	Sprint
UserStory1	As fleet manager I want to be able to log into (or logout) the system with my account at any time.	Done in Sprint 1.
UserStory2	As fleet manager I want to be able to add, modify or remove a bus.	Done in Sprint 1.
UserStory3	As fleet manager I want to be able to add, modify or remove a driver.	Done in Sprint 2.
UserStory4	As fleet manager I want to be able to view user requests on a map.	Done in Sprint 3.
UserStory5	As fleet manager I want to be able to view previous user requests.	Done in Sprint 3.
UserStory6	As fleet manager I want to be able to get the position of all the buses.	Done in Sprint 2.
UserStory7	As fleet manager I want to be able to get the utilization of all the routes.	Done in Sprint 3.
UserStory8	As fleet manager I want to be able to add or remove a route.	Done in Sprint 2.
UserStory9	As a bus driver I want to be able to log into (or logout) the system with my account at any time.	Done in Sprint 1.
UserStory10	As a bus driver I want to be able to see the schedule of the route I have to cover, with the user requests I need to satisfy.	Done in Sprint 2.

4.3 Nonfunctional Requirements

This section presents the nonfunctional requirements of our BusPlanner project, which describe the behavior of the system. We decided to divide them into 7 main categories.

4.3.1 Usability

- The application must be mobile responsive.
- The seat reservation should be done in the minimum number of steps possible.
- No fancy GUI.
- Correct and up to date information.
- Presenting data in a visible and understandable way.

4.3.2 External Libraries

- Our application makes usage of external libraries such as Bootstrap.

4.3.3 Compatibility issue

- The application is suggested to work only with the deployed version of the used libraries. Updated versions might bring incompatibilities.
- The maps being used should offer the possibility to work with PHP and SQL.

4.3.4 Security

- Only users of the application are allowed to use the project.
- The application needs to protect C.I.A elements (Confidentiality, Integrity and Availability) of user and nobody can see and change information of others.

4.3.5 Availability

Considering that the application:

- Can pave the way for users to take a bus as soon as possible with the aim of saving their time.
- Can make it easier for users to take a bus from everywhere in bus timetable.
- Can have friendly interface for users.
- Performance should provide the user a fast experience using the application.
- Has to handle user's request all the time using any device with an Internet connection and an installed web browser.

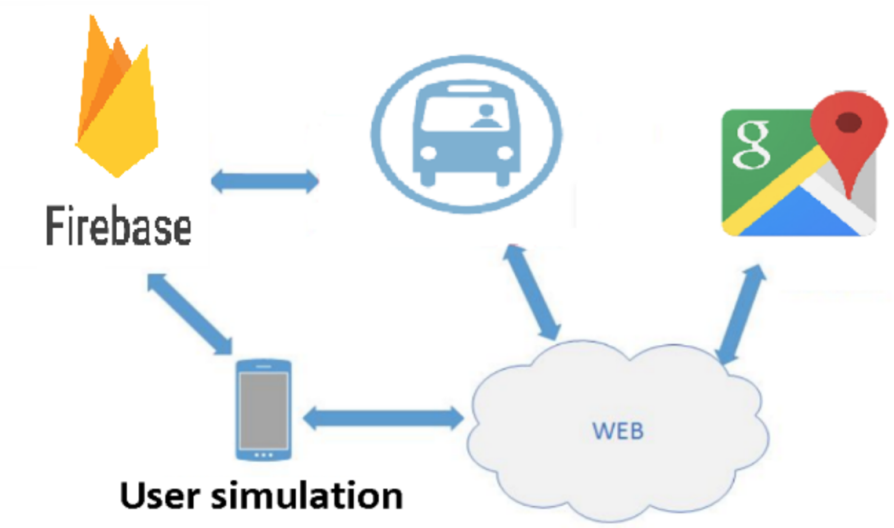
4.3.6 Uptime and data redundancy

The BusPlanner application should guarantee high availability and data redundancy. Still, since the application will be created in the context of the DSD course, our team will not build nor require any dedicated infrastructure for it and so estimating and proving exact value for data redundancy and uptime is not possible however, in the case 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.

4.3.7 Performances

The application has to be able to manage a high volume of requests. Since this application will be created in the context of the DSD course, our team will not build nor require any dedicated infrastructure for it. Furthermore, it is impossible to estimate and prove the exact value for performances. However, it should be easy to update it and improve it if needed.

4.4 High-Level Architecture



4.5 Technologies used

4.6 Algorithm

The algorithm is divided into static algorithm and dynamic algorithm.

4.6.1 Static Algorithm

It has been implemented by Andrea Colombo and Huy Hoang Nguyen.

The programming language used for its implementation was Python, with the addition of some libraries. It covers the simulation of user requests with the generation of timetables based on those requests. So, for the generation of timetables, the algorithm takes in input the user requests and produces the corresponding timetables.

The user requests are simulated through a Poisson distribution, based on the current time and a delta time that will then be added to cover all times of the day. The number of user requests for each time slot of the day depends on the lambda parameter of the Poisson distribution.

The intermediate passage is characterized by the grouping of user requests per route, that will later be sorted by time. Finally, the algorithm has to calculate all timetables elements (i.e. number of onboarding/deboarding passengers, departure and arrival time etc.).

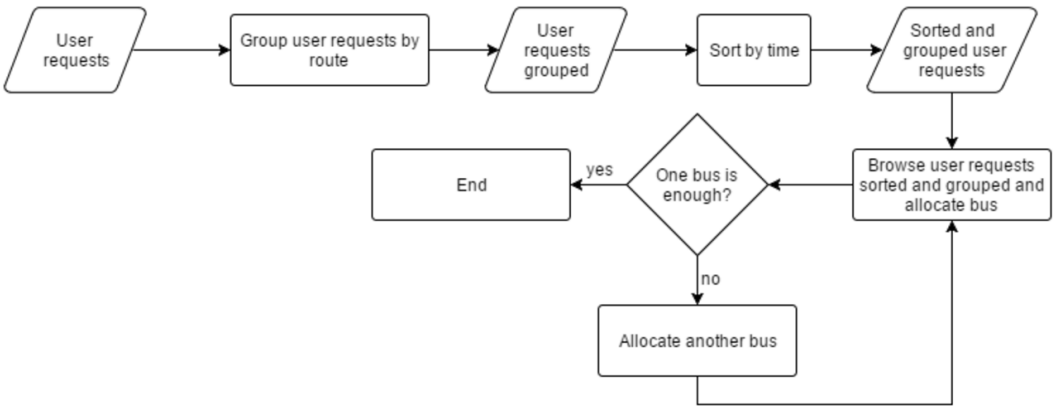


Figure 1: Static algorithm flowchart

4.6.2 Dynamic Algorithm

It has been implemented by Albi Dode and Sharvathul Hasan Ameerjan.

The programming language used for its implementation was Python, with the addition of some libraries and Linear Programming.

The initial idea was that we should have had one bus per route at a certain point of time, and each bus had to start and end at the bus depot.

The algorithm takes in input the number of onboarding and deboarding passengers and the starting and ending working hour of bus drivers, and produces as output the number of trips that a bus driver can do, the connection between time and trip and the route-bus assignment.

Since we had to implement a lot of requirements, which caused a lot of if & else statements, Linear Programming was used in order to fulfill certain criteria (named as rules in the code) and minimize some objective function.

The goal is to satisfy as many trips as possible in one duty.

5 DEVELOPMENT PROCESS

5.1 SCRUM Overview

The team has adopted the SCRUM software development methodology 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.
- It imposes a systematic-way-based approach through sprints, meetings and sprint backlogs.

5.2 Sprints

5.2.1 Sprint 1: 14/11 $\Rightarrow$ 28/11

- Fleet Manager can log into the system.
- Fleet Manager is able to manage buses.
- Bus Driver can log into the system.

5.2.2 Sprint 2: 28/11 $\Rightarrow$ 12/12

- Fleet Manager can manage drivers.
- Fleet Manager can get the buses position.
- Fleet Manager can manage routes.
- Bus Driver can view his personal schedule.

5.2.3 Sprint 3: 12/12 $\Rightarrow$ 10/01

- Fleet Manager can view all the pendant user requests on the map.
- Fleet Manager can view all the previous user requests.
- Fleet Manager can get the utilization of each route.
- The algorithm assigns buses to routes based on the requests.

5.3 Meetings

We did:

- Daily scrum meetings to assign tasks for the following days.
- Weekly meetings with supervisors (every Friday at 13:30).
- Weekly internal meetings right after the meetings with supervisors.

During the meetings we experienced a varying number of participants.

5.4 Issues with distributed development

The greatest challenge has been to coordinate people from different places. This because we experienced some lack of communication from some team members and, also, not all team members participated to the meetings.

During the meetings it was sometimes hard to understand each other because of the language.

Finally we had some difficulties in dividing the work for the algorithm and testing part. As s consequence, to complete in time the fixed requirements some members had to take care of activities delegated to other members.

5.5 Things we would change

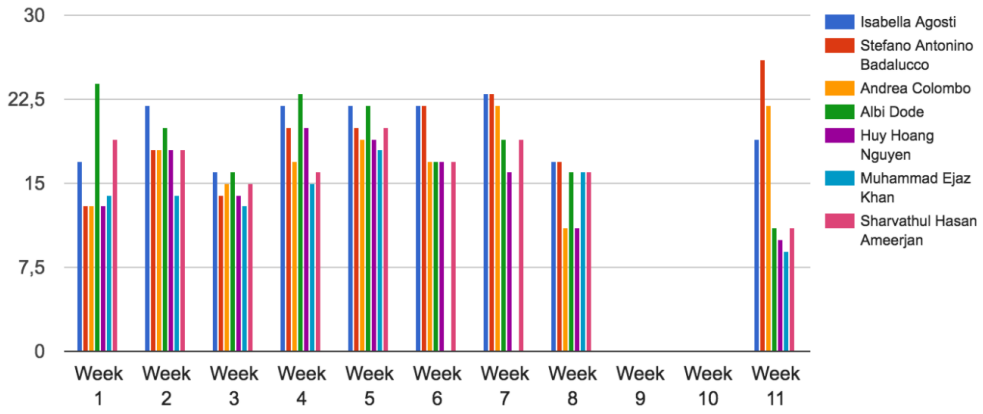
In order to avoid the issues that we encountered during the development of this project, we should have attended all the meetings, increased our communication and shared our work more frequently with the other group members.

Regarding the organization part, we should have worked more on the testing part, dedicated more time to the implementation and development part, rather than the documentation part and interacted more with the customer since the beginning.

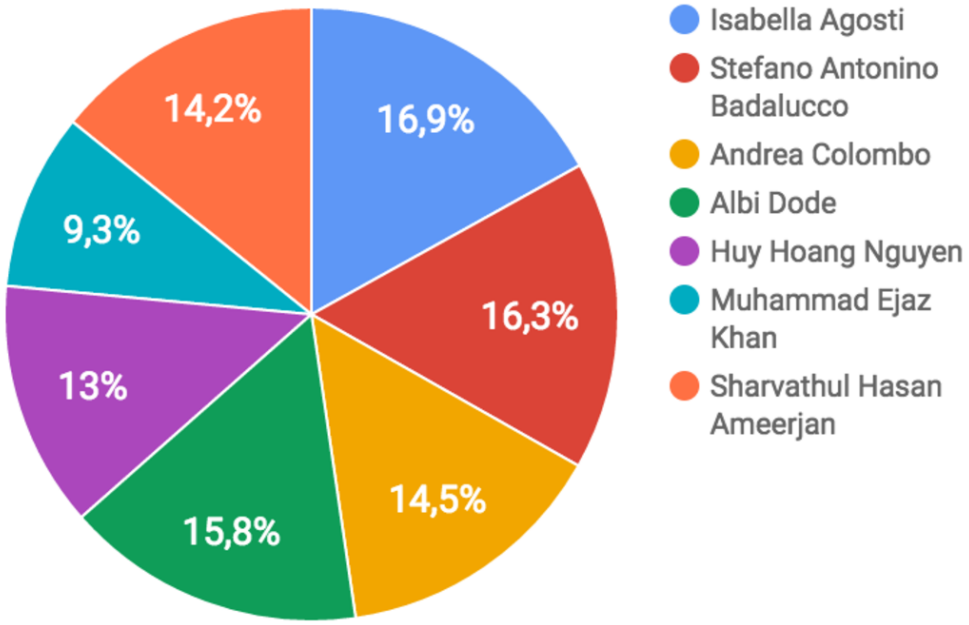
5.6 Process statistics

5.6.1 Working hours

Invested hours per person per week



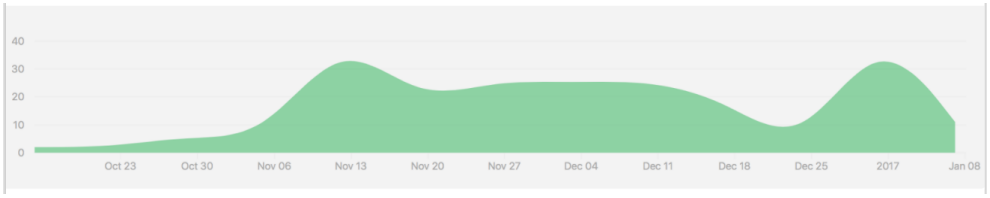
Total invested hours per person



5.6.2 GitHub contributions

Group contribution

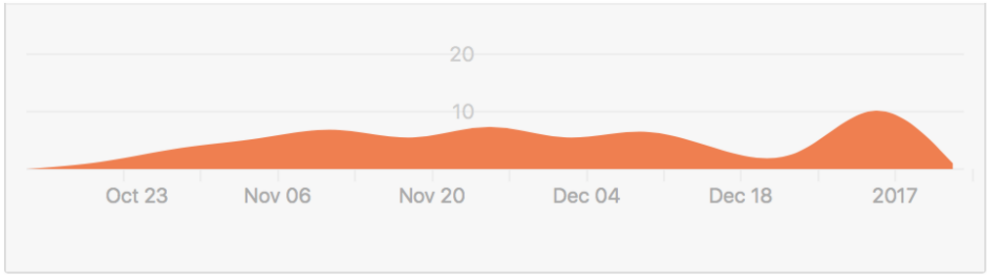
Total commits: **235**



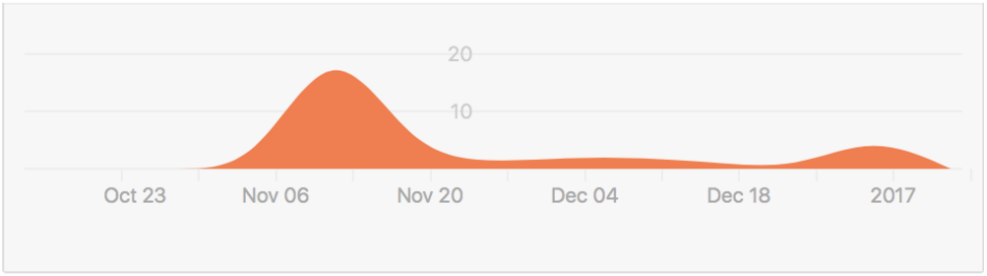
Individual contribution

- **Stefano Antonino Badalucco**: **72** commits.

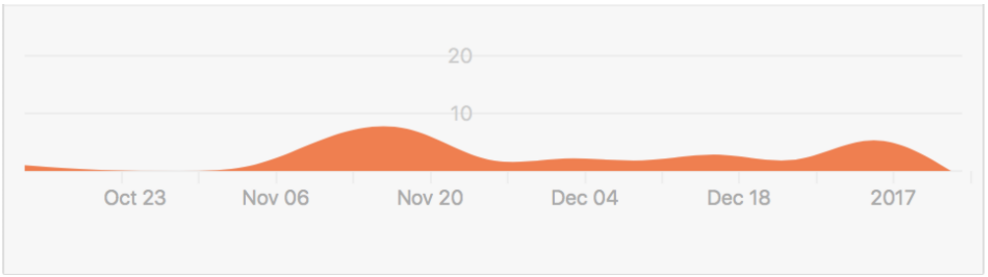
- **Isabella Agosti**: **62** commits.



- **Huy Hoang Nguyen**: **40** commits.



- **Albi Dode**: **33** commits.



- **Andrea Colombo**: **11** commits.

- **Muhammad Ejaz Khan**: **9** commits.

- **Sharvathul Hasan Ameerjan**: **2** commits.