ARTESIS PLANTIJN HOGESCHOOL ANTWERPEN

TryCatchUs

Professional Bachelor Applied Informatics

Technical documentation

Artesis Plantijn Hogeschool

Part of the informatics project supported by

Team8 - TryCatchUs

Begeleider: Dirk Duivel
Mentor: Inge Engel

Academiejaar 2019-2020
2de semester

Table of content

VERSION CONTROL

Nr.	Date	Distribution	Status	Changes
0.01	2020-06-03	Boamah Mark El Bahri Hamza Masri Abdulhadi Menouny Hamza Saket Kotiba	First draft	Summary of the project + impact on the infrastructure + release plan
0.02	2020-06-05	Boamah Mark El Bahri Hamza Masri Abdulhadi Menouny Hamza Saket Kotiba	Second draft	Techinical design + external system interfaces + data migration
1.00	2020-06-11	Boamah Mark El Bahri Hamza Masri Abdulhadi Menouny Hamza Saket Kotiba	Final version	Security and authorization + documentation + sources

TERMS AND ABBREVIATIONS

Term	Description
AP	Artesis Plantijn Hogeschool
IDE	Integrated Development Environment: a software that supports programmers to write code.
DFD	Data Flow Diagram: A diagram that shows the data flow between one or more system.
Azure AD	Azure Active Directory
OS	Operating System

SUMMARY OF THE PROJECT

The current situation at AP University College is that there is no computer system to keep track of reports of faulty facilities or needed services. AP employees report to the administrator if there is a defect. Program directors also must report to the administrator when they need certain services for their programs. This information is passed on to the administrator either by contact or by email. Most of the times, employees might have already forgotten about the details of the defect by the time they meet the administrator in person. If these reports are sent by mail, they are not properly organized for the administrator to keep track of. This makes most defects not attended to on time.

The aim of this project is to centralize, in addition to all communication between AP employees and service providers, in one system, also to solve the problems that arise with the current procedures. In this way, communication can proceed directly and, since all data is available to everyone, redundancy and ambiguities are avoided. Communication mainly consists of reports of defects and various assignments, including redesigning rooms. It is important that this system is always accessible on all platforms used by the facility services, logistics services and other employees. Therefore, a web application will be provided.

Defects can be reported intuitively. Furthermore, it will be possible to submit assignments on both applications ranging from refurbishments to projector installations. In addition, it will also be possible to request both current and archived reports.

IMPACT ON THE INFRASTRUCTURE

Our project will require a certain infrastructure. We briefly discuss below the impact on the current infrastructure and the requirements of the future infrastructure.

CURRENT INFRASTRUCTURE

The current procedure for reporting a defect does not require any infrastructure, since it is communicated orally; by telephone or by email. As a result, no infrastructure is available.

This makes it very easy to choose and implement a new infrastructure in terms of software and hardware, because we can start with a clean slate.

FUTURE INFRASTRUCTURE

FacilityTool will be hosted on a server on which few to many infrastructures will run. Thus, all requirements for these services will have to be taken into account, including the OS and the requirements for Facility Tool itself.

The AP-supplied server will run on CentOS 7 with the following minimum requirements.

CenOS 7	**20GB storage space**
MongoDB	256MB RAM
Facility Tool	50GB storage space 1GB RAM

RELEASE PLAN

FacilityTool is a web application and therefore no installation is needed. All you need is a web browser and you can access it on any device irresepective of the the operating system. The release of the application will be handled by AP University College.

For migration to the server, a runner has been installed on our production environment and linked to CI/CD of our project on GitLab. The runner runs the script defined in the ".gitlab-ci.yml" file in our project. The script pulls project from the master directory, builds it (from the script inside the docker-compose.yml) and deploys to the production environment. The build and deploy processes are automated due to the continuous integration and continuous delivery (CI/CD) of GitLab.

TECHNICAL DESIGN

Our project will have to rely on a number of technologies at every layer. We have carefully made our selection through in-depth comparisons between the largest technologies in the tech world. We have listed below the technologies we have chosen for frontend, backend and database.

FRONT-END TECHNOLOGY

As a front-end of Facility Tool app, Angular, a JavaScript framework, will be used. Angular is a development platform that aims to make web development feel effortless, focused on developer productivity, speed and testability. Applications built with Angular can be deployed to mobile devices and desktops as websites and native applications.

Angular is the frontend part of the MEAN stack, consisting of MongoDB database, Express.js web application server framework, Angular itself, and Node.js server runtime environment.

BACK-END TECHNOLOGY

We will be using Spring boot for the backend. Spring boot is a Java framework. Java is a cross-platform programming language, so the server's operating system does not matter for development. Spring Boot's framework makes developing and testing Java projects easier.

DATABASE

SQL VS NOSQL

The most important choice in the field of databases is between a relational or non-relational database. We eventually opted for a non-relational (NoSQL) database. The reason for this is that we can store the objects we work with directly in the database, without having to convert everything into tables. This makes it much more flexible and during development we can easily choose to store more or less data without having to create a schedule to suit it.

MONGODB

The database we have chosen is MongoDB. This is by far the most popular NoSQL database and has the advantage of offering very good support for all major programming languages.

Furthermore, the documentation is very complete and there is always enough information on the internet in case we encounter an unforeseen contingency.

VERSION CONTROL

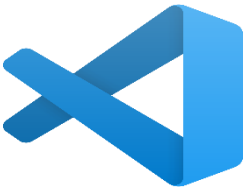
Gitlab will be used for version management. GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and continuous integration/continuous deployment pipeline features, using an open-source license, developed by GitLab Inc.

IDE'S

INTELLIJ IDEA

Facility Tool's backend will be developed with JetBrains' IntelliJ IDEA. This IDE is specially developed for Java, but also supports other languages. Furthermore, IntelliJ IDEA has an impressive code completion that makes development a lot more pleasant and faster. The built-in Git functionality ensures that you do not have to leave the IDE to commit code changes, for example.

VISUAL STUDIO CODE



Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

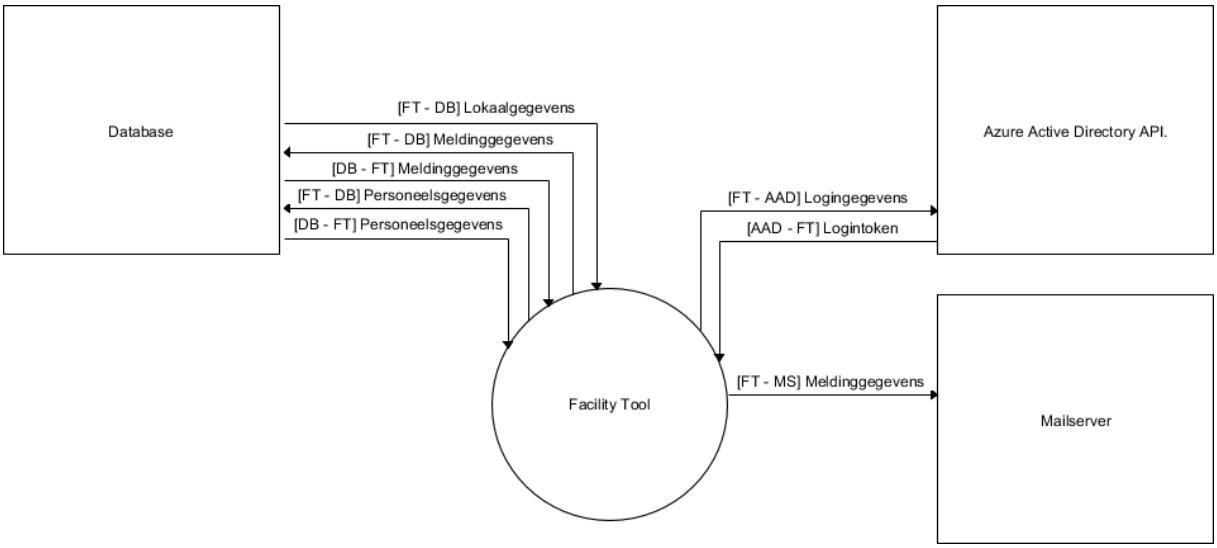
EXTERNAL SYSTEM INTERFACES

In order for the functionalities within our web application to work, it is necessary to be able to communicate with other systems. Interfaces are used for this. In this chapter, we describe which interfaces are needed and how they communicate with our system.

MICROSOFT AZURE ACTIVE DIRECTORY

To be able to use FacilityTool, all employees must be able to log in as the first step. For this we will use an Azure Active Directory API. Through this API we can login users with their unique access token given by OpenID Connect.

Below is the DFD that illustrates what data is exchanged when a user logs in. When the user enters his credentials, the Azure portal checks whether it matches what is in the Active Directory. After this, if it matches, a confirmation will be returned and the user is logged in.



All other information, such as users who report a defect or submit an order, is stored in our own database. Therefore no other systems are used for this interfaces. This data is therefore not included in the data flow diagram or DFD.

DATA MIGRATION

There will be no data migration in this project. This is because there is currently no system to keep track of reports or orders. As a result, we have to either start with an empty system or to manually place all current reports (from mails and paperworks) in the system.

SECURITY AND AUTHORIZATION

In our web application, different roles have been defined, each of which has access to certain actions. Therefore, in order to perform these actions, they must log in to our system. Security is used for this to have control over who gets access to the system.

SECURITY

OpenID Connect generates and authenticates an access token for log-ins to allow the end user access to the client application. Azure AD will retrieve the access token and ensures that the data sent is encrypted. It prevents someone from just getting an access token and using it to log in.

AUTHORIZATION ROLES

After a user is logged in, there are seven possible roles that he can have, namely: administrator, facility coordinator, facility employee, logistics coordinator, head of training and employee. In addition, there is the option for external parties to view notifications and change their status without logging in.

	MW	LMW	FMW	LC	FC	OH	Admin	Ex
Aanmaken defect	X	X	X	X	X	X	X	
Aanmaken taak				X		X	X	
Eigen defecten bekijken	X		X		X	X	X	
Eigen taken bekijken		X		X		X	X	
Eigen meldingen wijzigen	X	X	X	X	X	X	X	
Eigen meldingen annuleren	X	X	X	X	X	X	X	
Reactie plaatsen	X	X	X	X	X	X	X	X
Noodnummer contacteren	X	X	X	X	X	X	X	
Archief bekijken	X	X	X	X	X	X	X	
Meldingen exporteren							X	
Archief exporteren							X	
(De)Abonneren op defecten	X	X	X	X	X	X	X	
Categoriebeheer							X	
Ex beheer							X	
Notificatiebeheer	X	X	X	X	X	X	X	
Rollenbeheer							X	
Noodcontact beheer							X	
Email sjabloon wijzigen							X	
Defecten toewijzen					X		X	
Taken toewijzen				X			X	
Defect status updaten					X		X	
Taak status updaten				X			X	

DOCUMENTATION

TECHNICAL DOCUMENTATION

When developing an application it is important that good documentation is provided. This makes the aftercare of the application for future developers much easier because the operation of the code will be described in detail. Under normal circumstances, documentation is written using Microsoft Word, with a “functional” look. But we diecided to use a generated documentation since it adds “life” to the documentation in the sense that, the documentation automatically syncs with the latest updates.

For this reason, we will use Javadoc (for backend) and Compodoc (for frontend) in our project. The generated documentation is in html and since there are multiple pages, we thought it better to host the documentation online. This will even make accessing the documentation available at any time. Below is a link to the Compodoc documentation.

JAVADOC

Javadoc is a documentation generator. This creates HTML pages along with API documentation. It clearly shows how the Java application is structured. A hierarchy of the packages and classes is shown. You can clearly see which variables and methods are in the different classes and what they do in the application.

https://team8javadoc.imfast.io/JAVADOC/

COMPODOC

Compodoc is a documentation tool for Angular applications. The goal of the tool is to generate a documentation for all the common APIs of Angular: modules, components, injectables, routes, directives, pipes and classical classes.

https://team8trycatchus.imfast.io/documentation/

SOURCES

[1] What is compodoc?. https://medium.com/vincent-ogloblinsky/compodoc-documentation-tool-for-angular-2-applications-44ec650e01a8 (2020-06-11)

[2] GitLab?. https://en.wikipedia.org/wiki/GitLab (2020-06-11)

[3] Visual studio code?. https://code.visualstudio.com/ (2020-06-11)