**Research Pipeline CI/CD**

Ease of configuration

Gitlab CI/CD can be installed anywhere, on almost any type of system. This opens up the possibilities for any software developer because your operating system does not matter. You don’t even have to pay massive amounts of money in order to host this service. Another benefit is the ability to orchestrate it with Kubernetes (microservices). Hosting it using Docker is also a possibility, which is one of our criteria in order to make deployment easy. You’re less dependent on the hosting system.

Because of the fact that Gitlab is being hosted by Fontys, where we’re studying, it was the most logical place to begin with. The fact that our partners, Dex, are also Fontys students made it a logical choice for us to use GitLab. Our search for a way to realise a pipeline became quite easy as soon as we noticed that Gitlab has a built in pipeline.

Along with the fact that it is built in the hosting environment for our code, the DevOps maturity feedback is a nice feature. We are able to compare the quality of our pipeline with other companies. This way we can determine whether or not we need to implement changes to improve it and how. Thus making our code safer and of a better quality.

Deployment scheduling

Even though our current focus does not lie within deploying our application, we have taken into consideration that the tool we are going to use, offers us a nice way of deployment management. The Gitlab pipeline offers the ability to schedule whenever a branch should be released. We can avoid unnecessary problems for the end user this way by limiting big changes to the moment when traffic is at its lowest.

[**https://searchsoftwarequality.techtarget.com/video/5-advantages-of-GitLab-CI-CD-pipelines**](https://searchsoftwarequality.techtarget.com/video/5-advantages-of-GitLab-CI-CD-pipelines)

<https://medium.com/@sanjeevm/gitlab-pipelines-for-the-win-546e8477cb28>

**Research frontend: Entity Core Framework**

One of the key benefits Entity Core Framework offers is that programmers don’t have to design the data persistency layer: it does that for you. When you design the classes, Entity core has the ability to map the classes making it work directly with the database. This way the developers can completely focus on what’s important. Resources don’t have to be split and you save a lot of time like this.

Speaking of the database, it handles the connection to the database for you. You just have to determine which (SQL) database you’re connecting to. We do however need to take care of the security ourselves. This is one of the areas EF Core does not take care of for you. But when it comes down to SQL Injection, it does take care of that through the LinQ queries. This will render it safer and less prone to injection attacks.

Concurrent transactions will not be a problem. It handles that for you so you don’t have to make joint tables to retrieve multiple rows. This makes the code more readable.

https://www.partech.nl/nl/publicaties/2020/11/introduction-to-entity-framework#

**Research frontend: JS Framework**

What are the reasons for us to use Angular2?

There are a few reasons for us to use this framework. One of the main reasons is that DEX, the application we’re cooperating with, uses the same

Framework which makes integrating our application a lot easier.

Apart from that, the fact that it is using Typescript makes it easier to test. All the variables are strongly typed and thus easier to test. We can better expect what kind of output we receive. Compared to other loosely typed frameworks (which use vanilla JS) this is a lot more friendly towards test-driven code.

The fact that it is developed by Google also helps because there will be lots of documentation available for us to be able to use it. I assume google won’t abandon this project anytime soon so that implies long time support for this framework.

It is also highly scalable due to the way Angular functions (using components and modules). Instead of maintaining a monolithic application, we can just focus on a singular part without having to worry about breaking the rest of the application. If in any case a module or component needs to be added, you’ll just have to develop that one component and add it to wherever it needs to be.

Components made in Angular can also be recycled and repurposed for other modules or other parent components. You don’t have to write everything from scratch this way. It is just a matter of simply rearranging the parts and where you want them.

<https://angular.io/features>

<https://www.netguru.com/blog/benefits-of-angular>

<https://www.simform.com/blog/key-benefits-angular-use-cases/>

**Research backend: .NET Core Framework**

Due to the fact that this framework is compatible with all operating systems, it can run on any type of system. Any programmer can start working with the framework without having to worry about your workflow. It isn’t disruptive thus stimulating productivity.

.NET Core is also very lightweight and modular, making it work very well with containers. This way you can host the application across multiple servers. It also allows a mix of different technologies.

Open Source and community focussed, making it very dynamic and secure. Giving anyone and everyone access to the source code, ensures that it will work. And in the odd case that it doesn’t work, you have the ability to fix the problem yourself when one of the other hundreds of thousands developers doesn’t .

Built in dependency injection so we can change just one variable in the services for example and all the code can work. This makes the code interchangeable. It won’t break the code and you only have to change it in one place in order for the application to work.

Directed towards API development due to the fact that it offers a lightweight, high performance http performance request pipeline. Since we’re developing a Web API for DEX, Plex, this is a high priority.

Being architected for testability gives the developer the ability to write code and easily test the application. This way you reduce the risk of pushing an unstable build to production.

We also intended to make this application as transferable as possible for Dex because that will make the integration process as smooth as possible (might be unnecessary because it works through an API but familiarity is nevertheless a good benefit).

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/choose-aspnet-framework?view=aspnetcore-5.0>

<https://docs.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-5.0>

<https://www.zealousys.com/blog/benefits-of-asp-net-core-for-enterprise-application-development/>

<https://www.fortech.ro/top-advantages-net-core/>

**Transferability - What do we need to prepare to transfer our project properly**

***As I am going to pick up the new project, what do I need to be able to efficiently work on it?***

***Proper documentation***

***Which frontend framework are we using and why?***

***What programming language are we using?***

***What testing framework are we using for the frontend?***

***What did their workflow look like?***

***Which backend framework are they using?***

***What programming language are we using***

***What testing framework are we using for the backend?***

***What did their workflow look like?***

***Links to the repositories***

***What applications do I need to be able to run on my system in order to replicate their conditions?***

***What does the current state of the application do and what is it supposed to do?***

CI/CD pipeline setup

Testcases based on the user stories/use cases

Unittests

Integration tests

Functional requirements, properly documented

Put in the user stories

Feature Compass, detailed description of how to achieve the user story

UML Diagrams of the classes

UML Diagrams of the architecture

**Process diagram of the flow for the user (Business Diagrams)**

**Overview of tests and related user stories/cases**

Non functional requirements, properly documented

Describe which frameworks we’re using and why

frontend

backend

**CI/CD**

**Deployment**

What do they need to do to recreate our workflow

Link them to the resources we’ve used

Write instructions, enabling them in being productive

How can they open our documents and diagrams

Roadmap of how we approached the application

Sufficient research they can refer to and understand why we did what we did

Write down which conventions we’ve applied to our code

Comment in code whenever something needs to be elaborated

Accessibility of documents

One general place for them to retrieve files/documents

Example of folder layouts and where to find things

Getting started guide

Git Repo —> how to transfer? ***Fontys managed?*** Database information

Backend

Frontend

Credentials for Admin account or standard test user account

Git readme/wiki page