Research document back-end

CodexHub

Version table

Version	Date	Author(s)	Changes
1.0	March 2022	Florin Deleanu	First draft

Table of Contents

Introduction

The goal of the individual project for this semester is to create a full web application with a front-end and a back-end. In this document I will explain my choice of technology for the back-end side of the application.

The two most important things in my decision are if the chosen technology will benefit my future software engineering career and if it's a good fit for this semester. Other things to keep in mind while deciding what technology to use are as follows:

- Learning possibilities
- Popularity and job prospects
- Support of important functionality such as logging,authorization etc

The choices considered for this application were C# .NET Core, Java Spring Boot and Python Django/Flask.

Choice of technology

Java Spring Boot

Spring Boot (Tutorialspoint, 2022) is an open-source Java-based framework used to create a micro-Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications.

Being one of the most popular languages and frameworks for building web applications, many libraries and resources are available for studying purposes and choice of libraries. Its performance is slightly below .NET Core. (Prema, 2022). I have used this technology for building applications in a past project, so the learning opportunities are lower than the other considered choices.

C# .NET Core

The .NET Core is the evolution of Microsoft's .NET technology into a modular, cross platform, open source and cloud ready platform, which runs on Windows, Mac, Android, IoT and Linux.

While less popular than the other choices, it is still one of the top competitors in the market. Therefore, there are just as many libraries and information sources with the exception of very low-level specific usages.

C# .Net Core boasts very fast performance times and is suitable for building enterprise-level applications. This choice makes use of the Microsoft technology stack, so implementing new features and building a scalable architecture has clear guidelines.

I already have quite some knowledge about C# as I've used it during my internship, but I've never created an app from zero and implemented everything by myself as the application was already developed and I just added to it.

Python Django

Developers created the Django framework for rapid application development. And for that reason, Django brings in ready-to-use components to accomplish various programming tasks. The key reason Django became so popular is that it brings almost everything out of the box to create a fully functional web application in a matter of hours. (Eduwyre, 2022)

Django brings in a very modular architecture, and various components are decoupled from each other. It also provides rules on how to implement other parts of the app such as the database which cannot be ignored (semi-opinionated). Because of this, using a NoSQL database is difficult and needs to modify the default ORM which is discouraged.

The existing prototype is also written in Python which makes this choice suitable for integration purposes. On top of that, Django is immensely scalable, as it is capable of meeting massive traffic demands, it is pluggable, meaning that developers can use plugins to extend web applications and there exists hundreds of packages available for lots of purposes. Moreover, Django is highly secure as the framework has protection against security attacks like XSS and CSRF attacks, SQL injections, Clickjacking, etc.

Python Flask

Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. (Wikipedia, 2022)

Because it's a micro framework, it's not as suitable as other alternatives to building enterprise applications. It also places more responsibility on the user to choose the correct and up to date library, technology etc. as there are many possibilities and no default choices. This, in my eyes, is a big disadvantage.

Initially, I wanted to combine Flask and .NET to learn more, but I changed my mind and dropped the idea because of the reasons listed above. Furthermore, building a complete app in the .NET software stack has advantages because there are many possibilities to reuse code such as creating NuGet packages and also all of the microservices can make use of the standard supported options offered by Microsoft.

Conclusion

All the choices discussed can be used to build enterprise level applications and more or less suitable for this semester.

The performance of all 3 languages is fast so the performance of the back end will mostly be influenced by the implementation of the logic that interacts with the database. Also, in a microservices type application, the API gateway is usually the bottleneck and not the individual microservices as they can be scaled separate from one another.

The syntax of C# and Java are remarkably similar, while Python is known for being simpler and more user-friendly. They are all also mature technologies with huge communities backing them so finding information and resources to use will not be a problem.

In the future I would like to pursue a back-end software engineering career so deepening my knowledge and filling in missing information would be very beneficial to me. Because of this reason and the explanations above, I choose .NET Core as the back-end for this project.

Bibliography

Joy, A. (2022). Retrieved from https://pythonistaplanet.com/advantages-of-django/

Eduwyre. (2022). Retrieved from https://eduwyre.com/article/spring-vs-django-differences-and-similarities

Prema, P. (2022). Retrieved from https://medium.com/@putuprema/spring-boot-vs-asp-net-core-a-showdown-1d38b89c6c2d

Tutorialspoint. (2022). Retrieved from https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm

Wikipedia. (2022). Retrieved from https://en.wikipedia.org/wiki/Flask_(web_framework)