.

Dedication

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Dedication

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa. .

Acknowledgments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras

viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Table of Contents

List of Figures

List of Tables

End of Studies Project

Anis, Benna
anis.benna@incedo.com

Firas, Jaber
firas.jaber@incedo.com

March 2022

General Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Chapter 1

Context of the work

Introduction

This chapter introduces the general context of this report. We start by presenting the frame of the project as well as the host company. Then comes the enumeration of the problems which led to the realization of the project. We wrap it up by defining the methodology we've followed to carry out our work.

1.1 General framework of the internship

This project was carried out within the frame of obtaining a bachelor's degree in Computer Science at the Higher Institute of Technological studies of Nabeul. The internship took place fully remotely at Incedo Services GmbH for five months starting from the 1st of February 2022 to the 30th of June 2022 with the purpose of further developing an existing project of the company as well as migrating it to a micro-services architecture deployed on a self-managed instance of Kubernetes.

1.2 Company overview

This section introduces the host company and as well as the services it offers.

1.2.1 About Incedo

We are a young software development and consulting firm located in Stuttgart. We help our clients to develop exciting digital products and solutions and we also love to bring our own ideas into life from time to time.

1.2.2 Incedo's services

Consulting

Incedo helps its clients overcome two main challenges:

- Develop a product that somebody actually needs and that creates enough value to form a profitable business case.
- Ensuring that (large) IT-projects are finished in time & budget with a result that satisfies the actual users/clients of the developed software solution.

Development

Incedo develops software for clients, as well as for the company itself for all sorts of clients.

Figure 1: Selection of Incedo clients

1.3 Stating the problem

Incedo -having ambitious goals to grow over the next 2 to 4 years- wants to win new clients and strategically develop the existing ones. Winning new clients starts with generating new leads. Previously, Incedo has worked with an Austrian start-up (motion group) that provides automated lead generation through LinkedIn at the cost of 0.85 € per requested contact. Although one big project was closed and several leads were generated, Incedo started using the LinkedIn Sales Navigator with a more targeted approach, but nevertheless wanted to automate the approach and increase its outreach. Having launched the first version of the ILG (Incedo Lead Generator) as a SaaS (Software as Service), Incedo was satisfied for a while. Over the time however, as more and more clients were interested in the ILG, the current architecture couldn't handle the load properly. Therefore, it was our task to re-design and implement a new scalable architecture instead of the old one.

1.4 Assessment of the case

1.4.1 Describing the work procedure

The work on any project must first of all be preceded by a thorough study of the existing ones which undermines the strengths and weaknesses of the current system, as well as the business decisions that should be taken into account during the conception as well as the realization.

1.4.2 Criticizing the current state

After studying the existing, we can determine its limitations:

- Bugs always tend to happen whenever the LinkedIn website changes (due to scraping).
- Since cron jobs are running for the whole day, bugs are hard to respond to fast enough because we can only deploy once at the end of day.
- It is hard to test the whole workflow because of how the app works (looking for certain changes in the LinkedIn interface after certain buttons are clicked for example) meaning that the dev environment is lacking.
- It cannot scale well enough since the whole application is deployed on a single server.

1.4.3 Proposed solution

The solution to these problems is refactoring the whole application to separate sub applications (microservices) where the automation and scraping processes can be scaled independently of the other parts of the application. This way, it will be easier to maintain and scale the different codebases as well as respond faster to bugs and know exactly what caused them in the first place.

1.5 Development Methodology

1.5.1 Agile methodology

Agile is a structured and iterative approach to project management and product development. It recognizes the volatility of product development, and provides a methodology for self-organizing teams to respond to change without going off the rails.

1.5.2 Scrum methodology

Scrum teams commit to completing an increment of work, which is potentially shippable, through set intervals called sprints. Their goal is to create learning loops to quickly gather and integrate customer feedback. Scrum teams adopt specific roles, create special artifacts, and hold regular ceremonies to keep things moving forward.

1.5.3 Kanban methodology

Kanban is all about visualizing your work, limiting work in progress, and maximizing efficiency (or flow). Kanban teams focus on reducing the time a project takes (or user story) from start to finish. They do this by using a kanban board and continuously improving their flow of work.

1.5.4 The choice for ILG

Kanban is based on a continuous workflow structure that keeps teams nimble and ready to adapt to changing priorities. Work items—represented by cards— are organized on a kanban board where they flow from one stage of the workflow(column) to the next. Common workflow stages are To Do, In Progress, In Review, Blocked, and Done. And since this project is very susceptible to changes from outside (LinkedIn), Kanban offered more flexibility than Scrum so that's why the team went with it instead.

Conclusion

In this chapter, we presented not only the host organization but also the general context of the project and why it's needed. We then criticized the current state of the ILG application and discussed the possible development methodologies. Lastly, we picked the most ideal choice for us which is Kanban.

Chapter 2

State of the art

Introduction

This chapter will present and study various concepts such as Automation, Web Scraping, Version Control, DevOps, State Machine with an emphasis on the DevOps terminology such as the CI/CD pipelines. We will talk about the most used tools in the market as well as which ones we settled on after making a comparison.

2.1 Automation And Web Scraping concepts

2.1.1 Automation

At its core, automation is leaving menial and recurring tasks to the automaton (or machine) in order to do more meaningful work as a human in the meantime. Implementing automation improves the efficiency, reliability and the speed of tasks that previously took humans a lot of time.

2.1.2 Web Scraping

Web scraping is the process of automatically extracting content and data from a website. Although data extraction is done in a brute way by reading texts from HTML elements, it is used in a variety of legitimate digital businesses like search engines, price comparison sites and market research companies. So contrary to what some might think, web scraping is completely legal.

2.1.3 Relationship between the two

Web scraping simplifies the process of extracting data and the automation process helps repeating the recurring task of extraction. As a result, the combination of scraping data, storing it and automating the whole process is getting very popular, especially with new technologies and tools arising in order to do just that.

2.2 DevOps

2.2.1 Definition

DevOps is the set of practices, techniques and tools used to speed up the Software Development Lifecycle (SDL) by bringing together two historically

separate functions of development and operations.

Development refers to writing code and software, whereas operations refer to provisioning servers, configuring them as well as deploying the apps to them, amongst other things.

DevOps teams focus on automating all of the above. The key terminologies around DevOps are Continuous Integration, Continuous Delivery and Infrastructure As Code (IAC).

2.2.2 Lifecycle

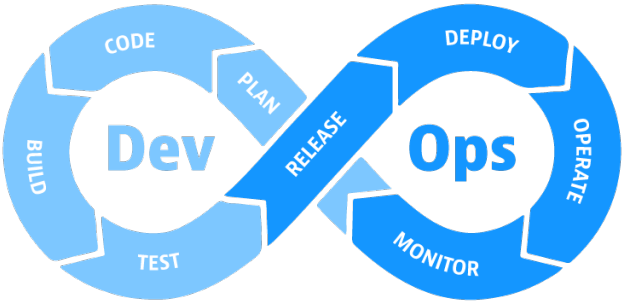


Figure 2: DevOps lifecycle

2.2.3 Container management

Containers have become an integral part of DevOps over the past couple of years but what is a container exactly and how do we manage them?

What is a container ?

Simply said; container is a packaging format for software applications that are akin to a very lightweight virtual machine which always executes in an isolated environment [1]. What this implies is that said containers can be easily copied to and run on different machines with high reliability, lower costs and high efficiency.

What is container management ?

Container management is the process for automating the creation, deployment and scaling of containers [3]. Container management tools such as Docker and Podman facilitate the addition, replacement and organization of containers.

2.3 Microservices

2.3.1 Definition

Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs. These services are owned by small, self-contained teams. [12]

Microservices architectures make applications easier to scale and faster to develop, enabling innovation and accelerating time-to-market for new features.

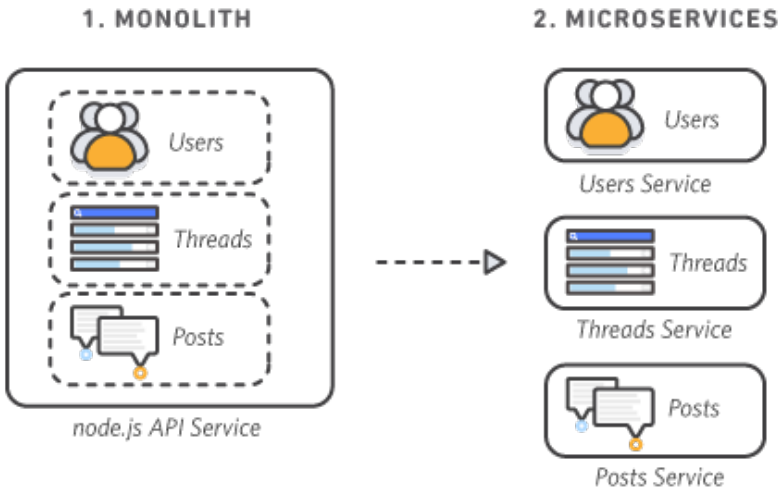


Figure 3: Breaking a monolithic application into microservices

2.3.2 Characteristics of Microservices

- **Autonomous :** Each component service in a microservices architecture can be developed, deployed, operated, and scaled without affecting the functioning of other services. Services do not need to share any of their code or implementation with other services. Any communication between individual components happens via well-defined APIs.
- **Specialized :** Each service is designed for a set of capabilities and focuses on solving a specific problem. If developers contribute more code to a service over time and the service becomes complex, it can be broken into smaller services.

2.3.3 Benefits of Microservices

- **Agility :** Microservices foster an organization of small, independent teams that take ownership of their services. Teams act within a small and well understood context, and are empowered to work more independently and more quickly.

- **Flexible Scaling :** Microservices allow each service to be independently scaled to meet demand for the application feature it supports. This enables teams to right-size infrastructure needs, accurately measure the cost of a feature, and maintain availability if a service experiences a spike in demand.

- **Easy Deployment :** Microservices enable continuous integration and continuous delivery, making it easy to try out new ideas and to roll back if something doesn't work. The low cost of failure enables experimentation, makes it easier to update code, and accelerates time-to-market for new features.

- **Technical Freedom :** Microservices architectures don't follow a "one size fits all" approach. Teams have the freedom to choose the best tool to solve their specific problems. As a consequence, teams building microservices can choose the best tool for each job.

- **Reusable Code :** Dividing software into small, well-defined modules enables teams to use functions for multiple purposes. A service written for a certain function can be used as a building block for another feature. This allows an application to bootstrap off itself, as developers can create new capabilities without writing code from scratch.

- **Resilience :** Service independence increases an application's resistance to failure. In a monolithic architecture, if a single component fails, it can cause the entire application to fail. With microservices, applications handle total service failure by degrading functionality and not crashing the entire application.

2.4 Comparative Analysis

In order to get started with our project, we need a wide range of tools that deal with the following areas; version control, orchestration between containers, browser automation as well as scheduled automation. For that, we did a comparative study on some of the tools the market provides.

2.4.1 Version Control: Git vs SVN

The most used tools for version control are Git and SVN. Here is a table comparing both tools.

Table 1: Comparative study between Git and SVN

	Description	Advantages
Git	Git is a distributed version control system which means that when coloning a repository, you get a copy of the entire history of that project.	Git has what's called a staging area. This means that even if you made over a 100 changes, they can be broken down to 10 commits each with their own comments and description.
SVN	SVN or Subversion is a centralized version control system. Meaning that there is always a single version of the repository that you checkout.	SVN has one central repository – which makes it easier for managers to have more of a top down approach to control, security, permissions, mirrors, and dumps.

At the end, both are great options. However, we will be using Git which is the VCS the company uses.

2.4.2 Container management: Docker vs Podman

Although Docker is widely popular, Podman is also taking its share from the market. Especially on Redhat systems.

Table 2: Comparative study between Docker and Podman

Aspect	Docker	Podman
Definition	Docker and Podman are both container management technologies used to build container images and store said images in a registry to then run them as containers in a target environment.	
Technology	Docker uses the containerd daemon which does the pulling of images then hands over the creation process to a low-level runtime named runc.	Podman uses a daemon-less approach using a technology named conmon which does the heavy lifting. It also delegates the container creation to a low-level container runtime.
Specificity	Docker Desktop is a great feature for Docker which provides an easy way to build and distribute containers amongst developers.	The smallest unit in Podman is the pod. A pod is the organizational unit for containers and is directly compatible with Kubernetes.

As a result, we will also be sticking to Docker in this project.

2.4.3 Browser automation: Puppeteer vs Playwright

Both are Node.js libraries for browser automation. The table below compares these two on different levels.

Table 3: Puppeteer vs Playwright highlights

Category	Puppeteer	Playwright
Overview	Puppeteer makes it easy to get started with browser automation. This is in part because of how it interfaces with the browser.	Playwright is very similar to Puppeteer in many respects. The API methods are identical in most cases, and Playwright also bundles compatible browsers by default.
Community	Has a large community with lots of active projects.	Small but active community.

Since the difference is pretty minor, we will be sticking to the more recent Playwright.

2.4.4 Database: MongoDB vs MySQL

The old architecture of ILG uses MySQL but we have the choice to use a different database as well, so we did the following study.

Table 4: Comparative study between MongoDB and MySQL [5]

Category	MongoDB	MySQL
Overview	An open-source NoSQL that stores data in JSON-like documents.	An open-source relational database system.
Storage	Stores data in documents that belong to a particular class or group.	Data is stored in tables where each table corresponds to an entity.
Type	NoSQL, meaning that data in a collection can have different structures.	Uses SQL (Standard Query Language) meaning that the schema of data cannot be changed once defined.

Since the old architecture uses MySQL, we will be sticking to that in the new architecture as well for an easier migration.

2.4.5 Database: GraphQL vs Rest

The most used tools for version control are Git and SVN. Here is a table comparing both tools.

Table 5: Comparative study between GraphQL and REST [4]

	Description	In depth
GraphQL	An open-source data query and manipulation language for APIs, and a runtime for fulfilling queries with existing data.	Describe only what's needed in queries, so no under or over fetching. It is also Schema and Type safe so it's less prone to bugs.
REST	REST (Representational State Transfer) is an architectural style that conforms to a set of constraints when developing web services.	It was introduced as a successor to SOAP APIs, follows the REST standards and is not constrained to XML.

Since the old architecture uses MySQL, we will be sticking to it in the new architecture.

Conclusion

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Chapter 3

Analysis and specification of requirements

Introduction

In this chapter, we are going to analyze the specification and the requirements of the application, mostly this chapter will contains the product main features and their conception so we will have our expectation for the end result.

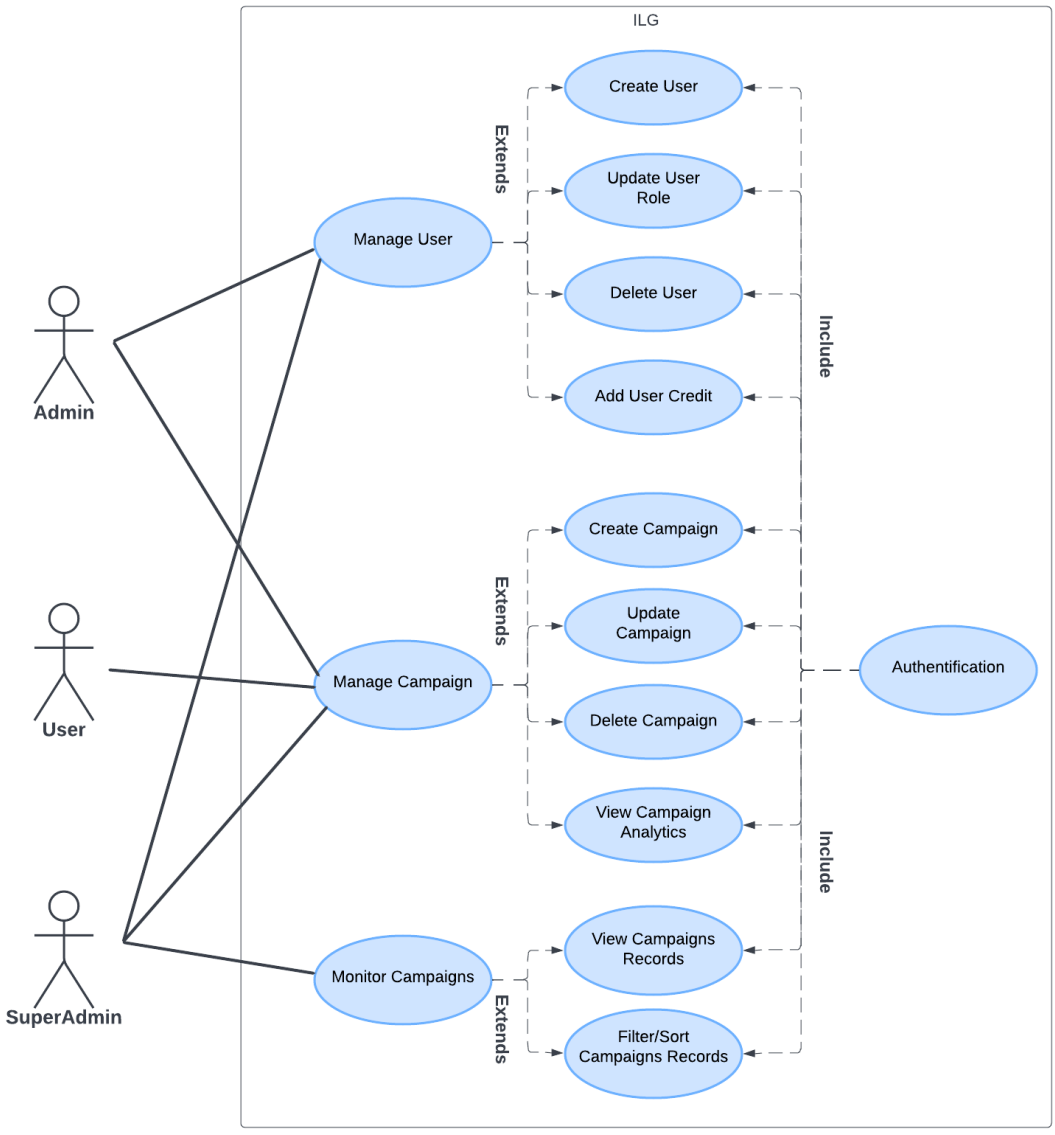
3.1 Requirements

Our requirements are divided into two parts, the clients portal of the application called Dashboard, and the lead generating and handling happening in the background.

3.1.1 Dashboard

3.1.1.1 Requirements

Here is a global use case diagram that specifies the dashboard requirements :



3.1.1.2 User management

3.1.1.3 Campaign management

3.1.1.4 Campaign monitoring

3.1.2 Lead generating

3.1.2.1 Requirements

3.1.2.2 Scraping leads

3.1.2.3 Contacting Leads

3.2 Conception

In this section, we will dive deep into the conception and the logic behind the parts of the application :

3.2.1 Architecture

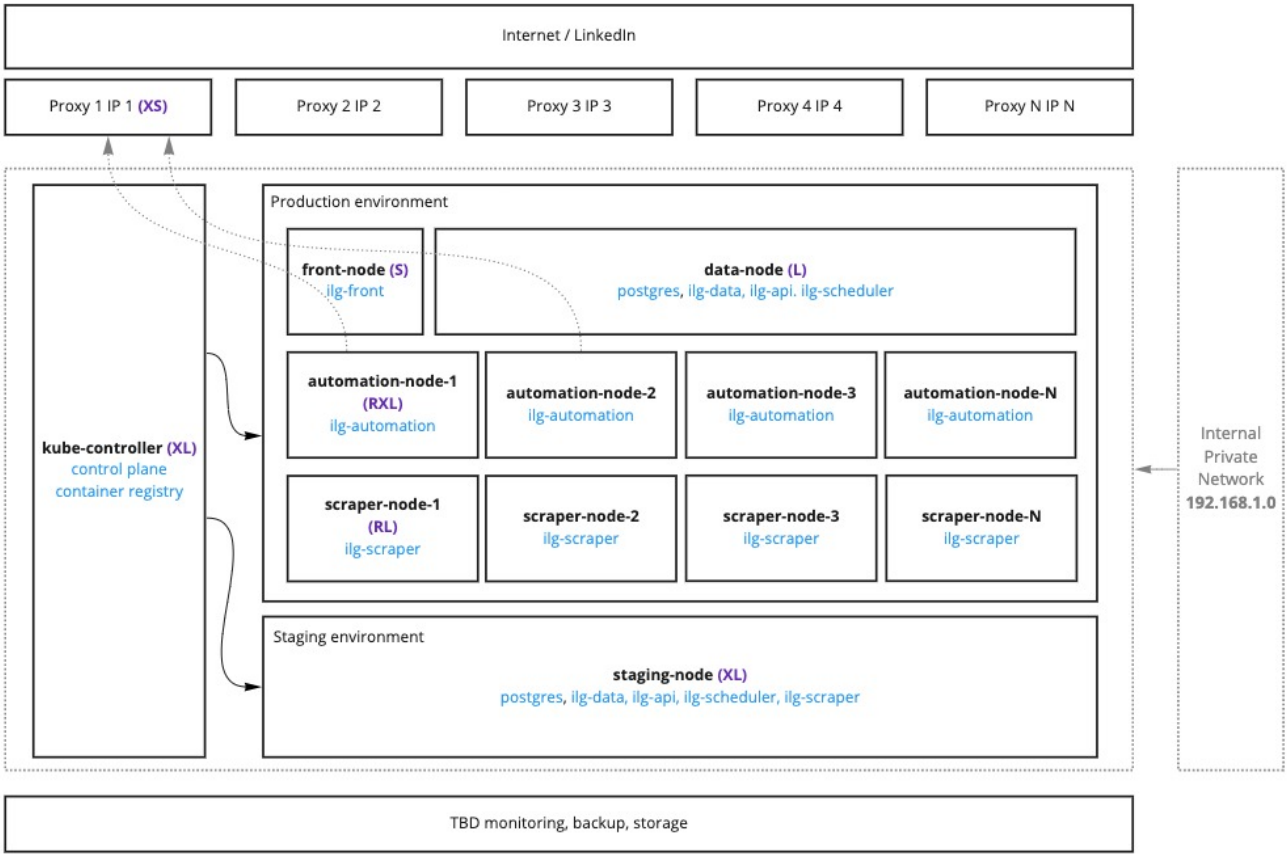
3.2.1.1 Microservices

Table 6: All the application's Microservices

Name	Description	Scalabity
ilg-data	GraphQL Database access layer used to feed all needed data for other services from the Postgres DB.	No
ilg-api	RESTful api to serve data to the React app dashboard.	No
ilg-front	a React app dashboard application served throw NGINX.	No
ilg-scheduler	Service thats responsible for scheduling and orchestrating the tasks queues and notifications.	No
ilg-scraper	Service thats responsible for scraping leads of campaigns links from LinkedIn Sales Navigator.	Yes
ilg-automation	Service thats responsible for handling LinkedIn accounts inboxs/threads and sendings invites to leads.	Yes

3.2.1.2 Cloud Infrastructure

Since we are using a Kubernetes Cluster for our Cloud Infrastructure, we decided to distribute our different microservices into their corresponding nodes (VMs) depending on their shared components and scalabity. So here is an overview of the current Cloud Infrastructure of the application :



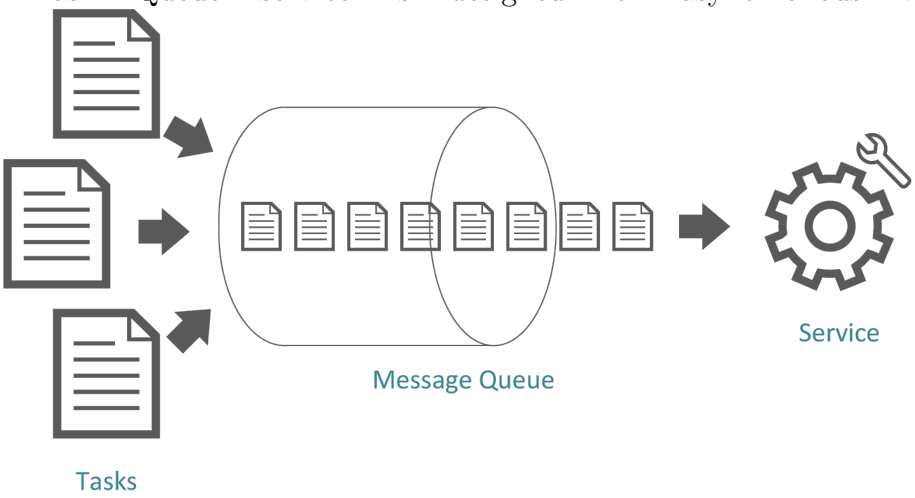
3.2.2 Dashboard

3.2.2.1 Conception of the features

3.2.3 Tasks & Queues

3.2.3.1 Thoery of tasks and queues

Task queues let applications perform work, called tasks, asynchronously outside of a user request. If an app needs to execute work in the background, it adds tasks to task queues. The tasks are executed later, by worker services. The Task Queue service is designed for asynchronous work.

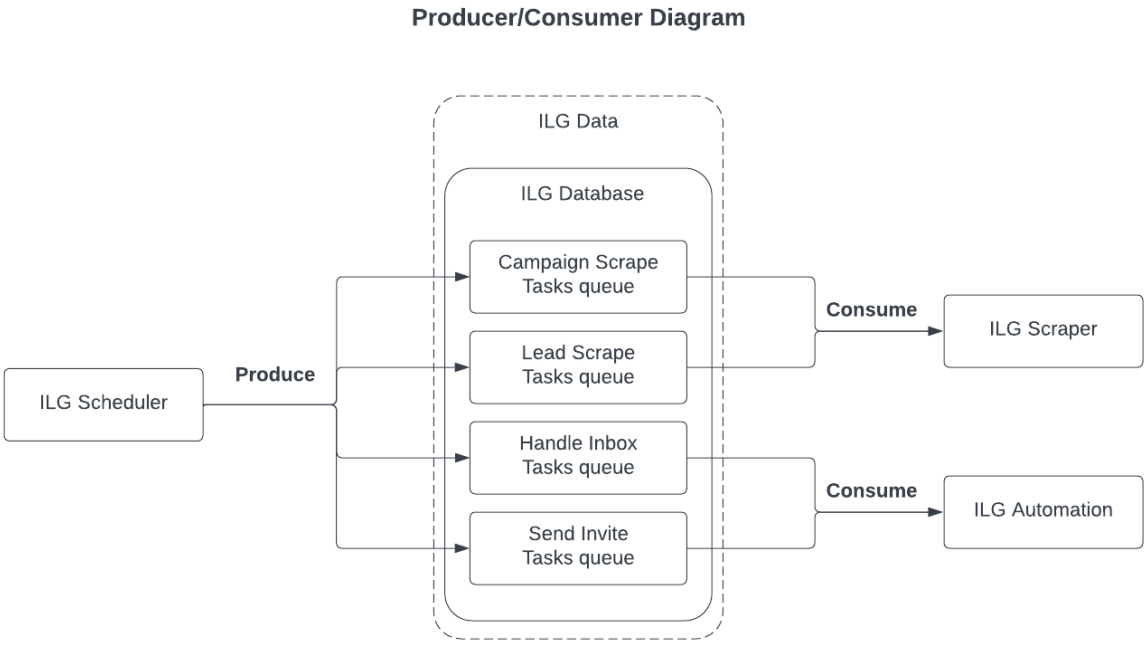


3.2.3.2 Tasks queues types

Table 7: All the application's tasks/jobs

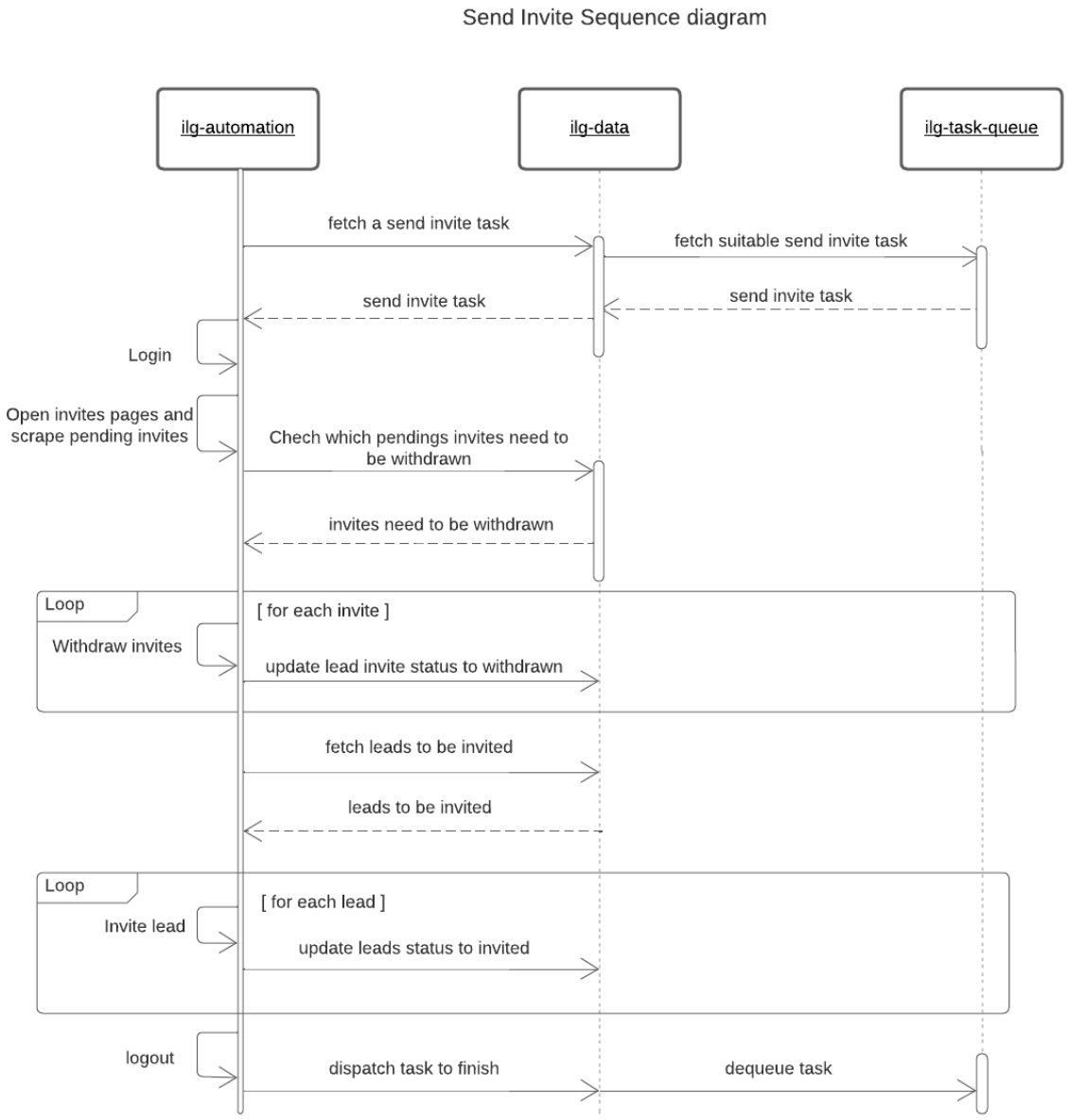
Name	Description	Producer	Consumer
campaign-scrape	Scrape list of leads from campaign search list	ilg-scheduler	ilg-scraper
lead-scrape	Scrape lead information and save them in the database	ilg-scheduler	ilg-scraper
handle-inbox	Hanle conversation threads of leads and send follow ups if needed	ilg-scheduler	ilg-automation
send-invite	Send connection invite to leads	ilg-scheduler	ilg-automation

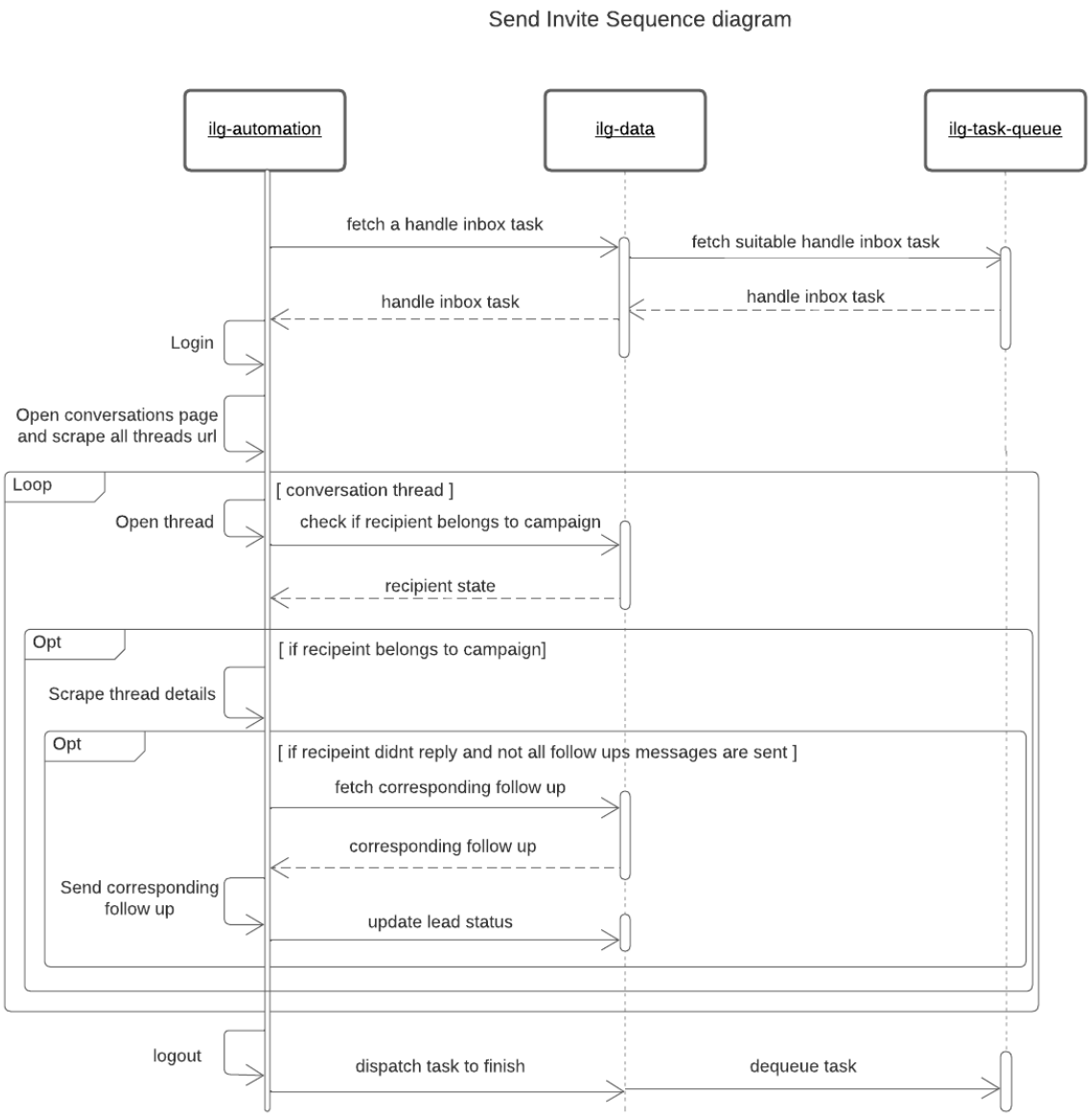
3.2.3.3 Scheduling tasks



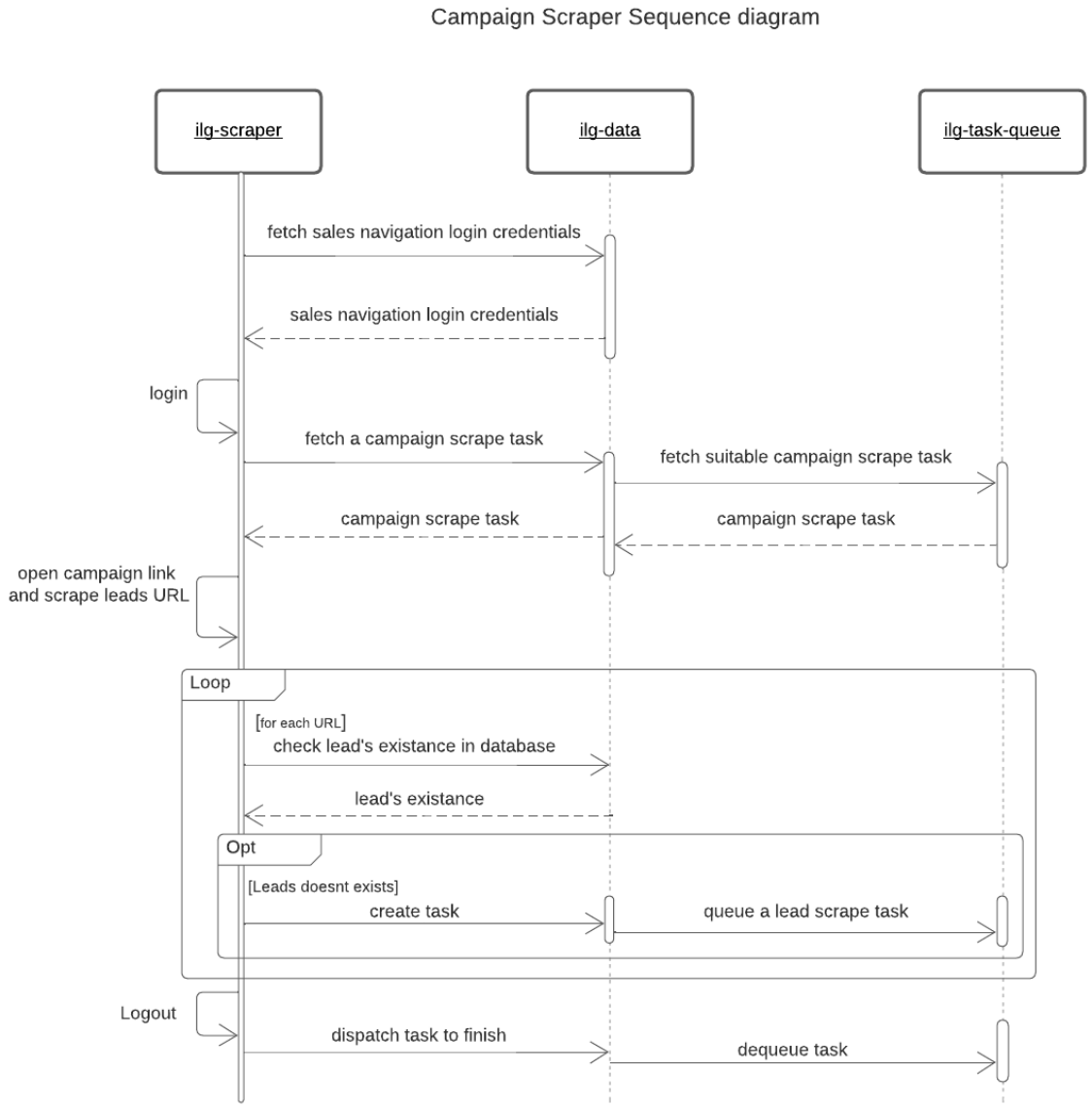
3.2.4 Automation & Scraper

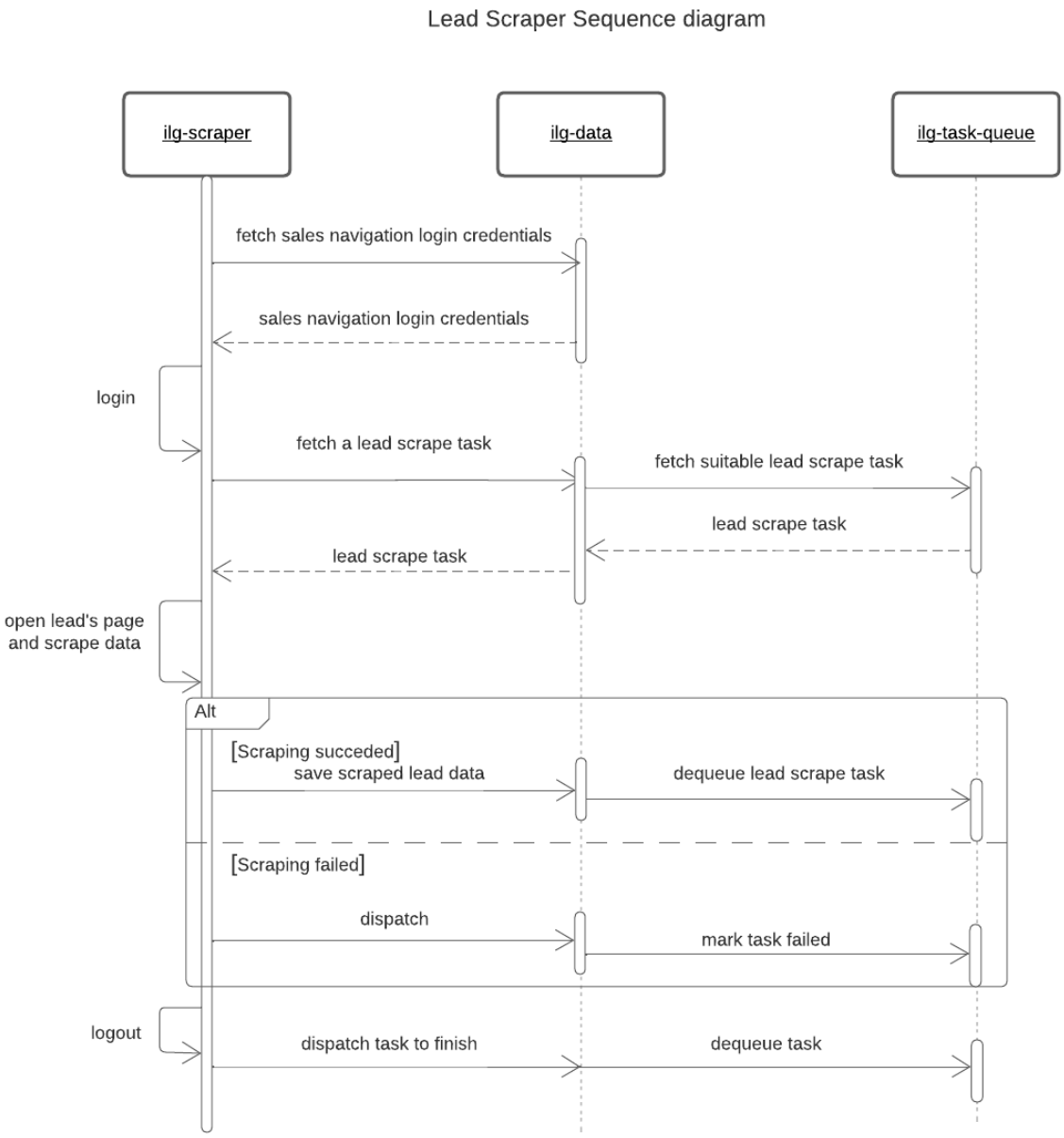
3.2.4.1 Automation flow




Send Invite Sequence diagram
ilg-automation
ilg-data
ilg-task-queue
fetch a handle inbox task
fetch suitable handle inbox task
handle inbox task
handle inbox task
Login
Open conversations page and scrape all threads url
Loop
[conversation thread]
Open thread
check if recipient belongs to campaign
recipient state
Opt
[if recipeint belongs to campaign]
Scrape thread details
Opt
[if recipeint didnt reply and not all follow ups messages are sent]
fetch corresponding follow up
corresponding follow up
Send corresponding follow up
update lead status
logout
dispatch task to finish
dequeue task

3.2.4.2 Scraper flow

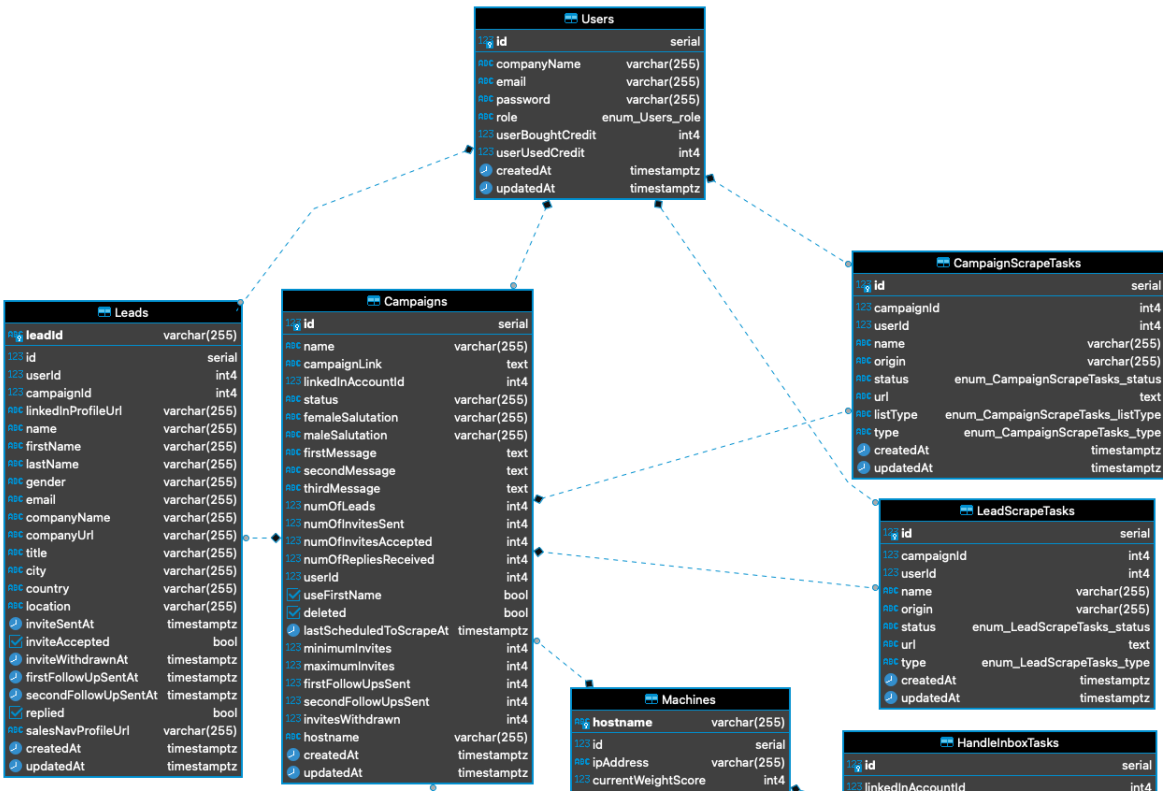




3.2.4.3 Horizontally Scaling

3.2.5 Data Layer API

3.2.5.1 Schema



3.2.5.2 Different interfaces

REST and GraphQL

Conclusion

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Chapter 4

Realization

Introduction

In this chapter, we will finally discuss the long awaited realization. This is the implementation phase where all of the work done in the previous chapters come into picture. We will start with the architecture where we setup the necessary hardware and software tools. Then we will present some of the difficulties we encountered.

4.1 Hardware environment

These are the servers or nodes that make up the Kubernetes cluster needed for the deployment. They can be separated into the core nodes that are created once, and the scalable nodes that can be created indefinitely.

4.1.1 Kubernetes Cluster

Table 8: Core cluster nodes

Hostname	Services	Size
kube-controller	The main Kubernetes cluster controller	XL
data-node	postgresdb, ilg-data, ilg-api, ilg-scheduler	L
front-node	ilg-front	S

Table 9: Scalable cluster nodes

Hostname	Services	Size
scraper-node-n	ilg-scraper	RL
automation-node-n	ilg-automation	RXL

The number of servers for **scraper**, and **automation** nodes will be scaled depending on the number of clients as mentioned in **Chapter 3: Analysis and specification of requirements**.

4.1.2 Proxy Servers

Table 10: List of proxy servers

Name	Services	Size
Germany 1	Upstream proxy server	XS

The two images below show the available VM sizes in IONOS cloudpanel.

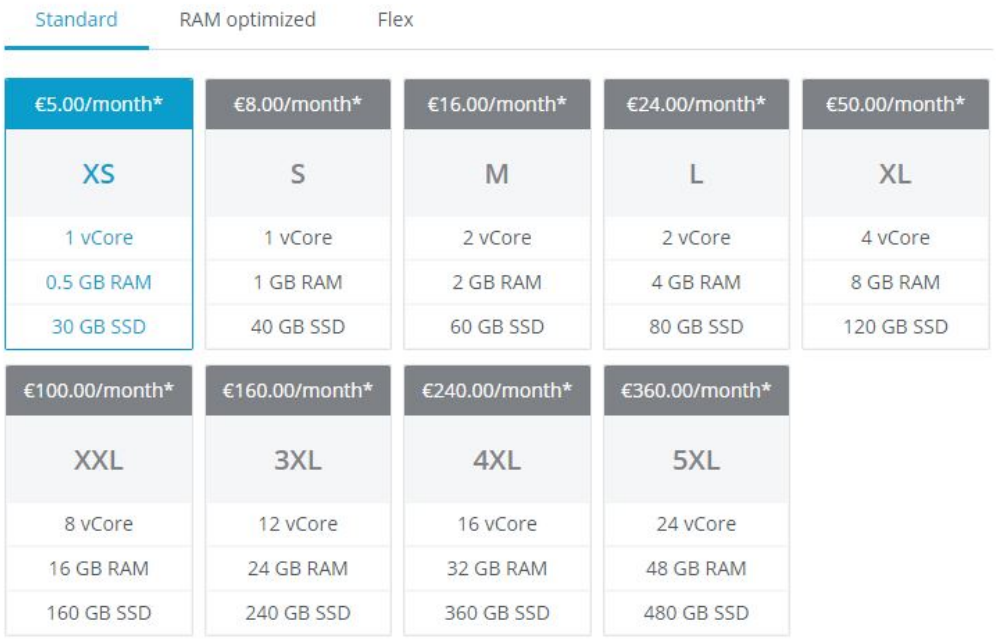


Figure 4: Standard VM sizes in IONOS

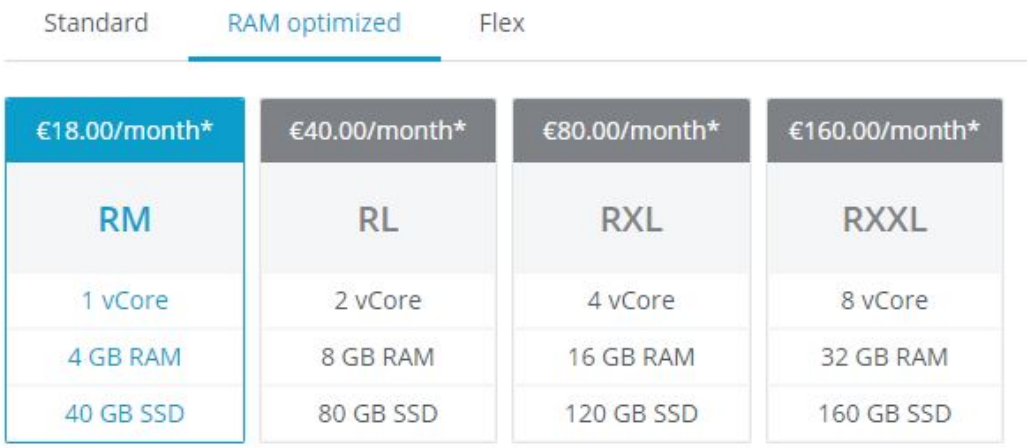


Figure 5: RAM Optimized VM sizes in IONOS

Also note that this section is only relevant in production.

4.2 Software environment

This part is reserved for the presentation of the software used in the realization of the application and includes but is not limited to; programming languages, frameworks, technologies, etc...

For a comparative analysis on the most crucial ones, see **Chapter 2: State of the art**.

- **Git:**

Figure 6: Logo of Git

- **Docker:**

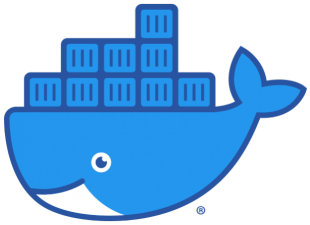


Figure 7: Logo of Docker

- **Playwright:**

Figure 8: Logo of Playwright

- **NestJS:**
  [10] A framework for building efficient, scalable Node.js web applications.

Figure 9: Logo of NestJS

- **ReactJS:**
  A front-end javascript library that's often used as a framework.

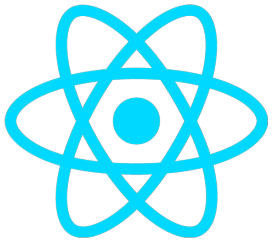


Figure 10: Logo of ReactJS

- **MySQL:**

Figure 11: Logo of MySQL

- **Sequelize:**
  [11] A modern TypeScript and Node.js ORM for SQL databases.

Figure 12: Logo of Sequelize

- **GraphQL:**
  [7] A query language for APIs and a runtime for fulfilling those queries with existing data.

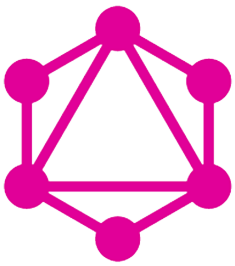


Figure 13: Logo of GraphQL

- **GitLab:**
  A DevOps software that allows secure collaboration and operations in a single application.

Figure 14: Logo of GitLab

- **Nginx:**
  [13] A multi-purpose web server can also be used as reverse proxy, load balancer, mail proxy and HTTP cache.

Figure 15: Logo of Nginx

- **Renovate:**
  A software that provides automatic dependency updates with support for multiple languages.

Figure 16: Logo of Renovate

- **Docker compose:**
  A helper tool to run applications with multiple Docker containers using a .YAML format. Used in development.

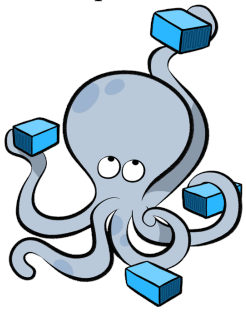


Figure 17: Logo of Docker Compose

- **Visual Studio Code:**
  A code editor that's used as an entire development environment with the help of extensions.

Figure 18: Logo of VSCode

- **Linux:**
  A Unix-like operating system for common daily use and more commonly for servers.

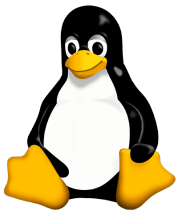


Figure 19: Logo of Linux

- **Make:**
  GNU Make is used extensively throughout the application to save useful commands for the ease of maintainability.

Figure 20: Logo of Makefile

- **MicroK8s:**
  [9] MicroK8s is a simple production-grade conformant K8s. It is Lightweight and focused.

Figure 21: Logo of MicroK8s

- **Kubernetes:**
  The most powerful tool for managing containerized workloads in the cloud.

Figure 22: Logo of Kubernetes

- **Ansible:**
  An open-source software for provisioning, configuring and managing infrastructure.

Figure 23: Logo of Ansible

- **Squid Proxy:**
  [2] Squid is a caching proxy for the Web that supports HTTP, HTTPS, FTP and more.

Figure 24: Logo of Squid Proxy

- **GitLab agent for Kubernetes:**
  [6] A secure and reliable way to attach a Kubernetes cluster to GitLab.

Figure 25: Logo of GitLab Agent for Kubernetes

- **LaTeX:**
  [8] A high-quality document preparation and typesetting system for technical grade documents.

Figure 26: Logo of The LaTeX Project

4.3 Difficulties encountered

This not persay a technical difficulty as it is more of a limitation from the side of the IONOS cloud provider the organization relies on. To have a full DevOps approach, even the infrastructure should be managed in code -also known as IAC (Infrastructure As Code)- using tools such as Terraform. However, the cloudpanel version that we were using does not provide that functionality. Therefore we had to scrape off the idea and simply create the virtual machines fron the UI everytime we need them.

Conclusion

A successfull project comes from a successfull development environment as well as a clean production environment. For this reason, we spent quite a good amount of time setting up the infrastructure in a clean and scalable way.

General Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Webography

[1] Arthur Busser. What is a container?, 2020. https://www.padok.fr/en/blog/container-docker-oci.

[2] Squid Cache. The official website, 2022. http://www.squid-cache.org/.

[3] Beth Pariseau Emily Mell. What is container management?, 2021. https://www.techtarget.com/searchitoperations/definition/container-management-software.

[4] Ronak Ganatra. Graphql vs rest apis, 2018. https://graphcms.com/blog/graphql-vs-rest-apis.

[5] GeeksForGeeks. Mongodb vs mysql, 2018. https://www.geeksforgeeks.org/mongodb-vs-mysql/.

[6] GitLab. The official website, 2021. https://about.gitlab.com/blog/2021/02/22/gitlab-kubernetes-agent-on-gitlab-com/.

[7] GraphQL. The official website, 2022. https://graphql.org/.

[8] The LaTeX Group. The official website, 2022. https://www.latex-project.org/.

[9] MicroK8s. The official website, 2022. https://microk8s.io/.

[10] Nest.js. The official website, 2022. https://nestjs.com/.

[11] Sequelize. The official website, 2022. https://sequelize.org/.

[12] Amazon Web Services. What are microservices?, 2022. https://aws.amazon.com/microservices/.

[13] Wikipedia. Nginx definition, 2022. https://en.wikipedia.org/wiki/Nginx.