BurgerHub

PA1489

Group 19

Alem Zvirkic(alzv23), Amir Hindawi(amhi22), Theodore Reangpusri(thre21), Tuva Rutberg(turu24)

Introduction Assignment 1

For this assignment we followed the instructions published on canvas.

Everyone is going to create their own menu, with burgers, side dishes, drinks and they get to name it as they want. We are going to split up the work into three areas of responsibility, Theodore is responsible for interface and github management, Alem is responsible for functionality and Amir is responsible for database and docker. Everyone has the responsibility to test the project by creating automated tests.

Timeplan

Below is our timeplan for the whole project. More detailed weekly plans can be viewed in our trello.

Week	38	39	40	41	42
To be done	Assignment 1 done	Assignment 2 done	Assignment 3 done	Summarize text and diaries	Complent. Turn in the project

It is possible to follow our weekly progress by accessing our Trello in which we have created an active issue board.

Bestäm utvecklingsmiljö

Developer Environments: Visual Studio Code, Docker
Revision Control: Git, Github
Coding Languages: Javascript, SQL, HTML, CSS
Frameworks: Tailwind
Project management and communication: Trello, Discord

We chose these tools and technologies because we have prior experience working with them, which helps streamline our development process. **Visual Studio Code** offers a lightweight yet powerful development environment with numerous extensions, while **Docker** ensures consistent environments across machines. For version control, **Git** and **GitHub** allow for easy collaboration

and tracking of code changes, which we've successfully used before. Our coding stack of **JavaScript, SQL, HTML, and CSS** is ideal for building a dynamic, data-driven web application, and **Tailwind CSS** helps us quickly create responsive, clean designs with minimal overhead. This combination allows us to work efficiently while leveraging industry-standard tools.

Bestäm git-server

We chose GitHub due to our previous experience with the platform and its widespread usage in the tech industry. Many major companies, including 90% of Fortune 100 businesses, rely on GitHub for collaboration and version control, which gives us a competitive advantage over others who might not be as familiar with Git. This familiarity ensures we can work efficiently from the start, leveraging GitHub's tools to manage our codebase effectively.

GitHub statistics

Skapa projektet

BurgerHub is an interactive web-based platform designed to enhance the burger ordering experience. It allows customers to browse and order a variety of burgers and side dishes directly from their devices. One of the standout features is the extensive customization options, where users can create their perfect burger by selecting from a wide range of ingredients, including various types of meats, vegetables, and other toppings. The site is intuitive and user-friendly, offering a seamless way to place and track orders, making it the go-to destination for burger enthusiasts looking for a personalized dining experience.

Katalog planering

We have created a github repository with all members and used git commands in the terminal in order to push and pull changes. More can be read in our diaries.

Git commands and reflections

• Vilka git-kommandon använder ni? Vad gör de?
• Vilka svårigheter stöter ni på? Hur löser ni dem?

Our diaries can be found in the *Reflektioner* folder which can be accessed via our github. The most common git commands that we use are:

Command	What it does	Comments
git fetch	Downloads objects and references from another repository without merging the changes.	We do this sometimes use this when we are unsure if we have all branches locally
git pull	Fetches from the remote repository and merges into the current branch.	We always start our session with pulling code from the main branch in order to not risk any potential merge conflicts.
git status	Shows the working tree status, indicating changes that are staged or unstaged	We try to use git status as much as possible.
git add	Stages changes for the next commit	We usually use git add . in order to add all files
git commit	Records changes to the repository with a descriptive message.	git commit -m “text” so send commit messages. We are keeping the messages short but informative
git stash	Temporarily saves your modified/staged files without committing them.	Useful for switching branches without committing.

git switch <branch name>	Switches branches in your working directory.	
git switch -c <branch name>	Creates a new branch and switches to it immediately.	

Initially we did use git init in order to actually use the git commands and push to github.
So far the only difficulties that we have faced are estimating how much time and effort that is needed for this project, how to work between branches without merge conflicts and initial setup for the actual codebase. By overcoming these by reading documentation and searching online for answers. We had a discussion regarding our prior experiences, strengths and weaknesses before creating the timeplan in order to estimate hours.

Sammanfattning av konfigurationshantering

Configuration management ensures consistency across environments, helping to prevent the "it works on my machine" problem. It provides version control for configurations, allowing teams to track changes and revert to previous states if needed. By automating the setup and management of environments, configuration management reduces the likelihood of human error and facilitates efficient scaling in complex infrastructures. Additionally, it enhances system reliability by maintaining stability and managing dependencies in a controlled manner.

Git arbetsflöde

The basic Git workflow involves having a main/master branch that contains the working version of the code. Collaborators create their own branches based on their assigned tasks, which allows them to isolate their work. This separation ensures that multiple developers can work independently without affecting the main branch. Once changes are made, you commit your work with a descriptive message, helping others understand what has been updated.

After committing, you push your changes to the remote branch, preventing any direct changes to the main branch. On GitHub (or other platforms), a pull request (PR) is created to review these changes. This ensures that the proposed changes are reviewed and do not unintentionally overwrite or remove important code. When a PR is created, Git automatically checks for merge conflicts, but it’s still a good practice to manually review the changes before merging.

Once a merge is complete, it's encouraged that all team members pull the latest version into their local main branch to ensure everyone is working with the most up-to-date code.

Våra erfarenheter av att arbeta med konfigurationshantering

Vad gick bra? Vad gick mindre bra? Hur löste ni svårigheterna?

GitHub
Working with GitHub was mostly smooth, with only minor complications. One group member experienced issues with Git credentials not authorizing login. This was resolved by switching the Git clone method to SSH. Another challenge we faced was when renaming files from uppercase to lowercase, which caused problems with pushing changes and getting Git to recognize that the old files no longer existed. We resolved this by using the command git rm --cached <old_filename> to cache the old files and remove them from Git tracking.

Docker
Working with Docker posed more challenges. Initially, everything worked well, aside from minor issues getting NPM to install the required modules to run the program. We solved this by ensuring that the package.json file included all necessary dependencies and by running the npm install command inside the Dockerfile.

The biggest issue we faced, however, was incorporating the database. Docker wasn't running the SQL files in the correct order, leading to an empty database. Debugging revealed two key issues. First, there was a pathing error that prevented Docker from locating the SQL files. Once the pathing was fixed, we encountered the second problem: the files were not executed in the correct order, causing the database to reset every time. We resolved this by renaming the SQL files in sequential order, such as 01-setup.sql, ensuring they were processed correctly.

After resolving these issues, we manually tested the database by running docker-compose exec mysql-db bash followed by mysql -u root -p, allowing us to access the database. While the database structure was created correctly, the data insertions had not worked, leaving the tables empty. Upon manually running the insert file (mysql -u root -p <file-name>), we discovered that local-infile=1 was not allowed. Running the command as mysql --local-infile=1 -u root -p <file-name> resolved the issue, which indicated we needed to enable local-infile=1 inside Docker.

We attempted to add this command directly into the Docker configuration, but it did not seem to have any effect. After further experimentation, we found that creating a cnf file with the necessary command to allow local-infile=1 fixed the problem. This worked until we tried running Docker from a Windows-based terminal (GitBash), where Docker could not execute the cnf file because it was seen as writable, not just readable. This issue was resolved locally by running chmod a-w <filename>. At first, it was unclear why the solution did not work

consistently, but we eventually realized the difference was in the terminals used to run Docker—it worked on Linux and Mac-based systems (Ubuntu and macOS) but not on Windows-based GitBash.

Hade ni kunnat göra annorlunda?

We could have been more thorough when debugging. Instead of fully exploring minor changes, we quickly abandoned them, which ultimately wasted more time. Later, we realized that the small adjustments we were making were on the right track. By combining these changes and analyzing them more carefully, we could have saved a significant amount of time

Vad lyckades ni inte göra? Varför inte?

We did not fully adhere to our planned schedule. The main reason was that getting Docker to work as intended took more time and effort than anticipated, which slowed down our workflow. This miscalculation occurred because it was our first time working with configuration management; however it will now be easier to estimate effort and time taken for future tasks.