BurgerHub

PA1489

Group 19

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

2. Länk till projektets sida på git-servern.

GitHub

3. Kort sammanfattning om vad ni implementerat. Beskriv med 5–10 meningar

The project we worked on is a burger ordering system called BurgerHub, designed to allow users to customize and place orders for burgers, side dishes, and drinks. The system features a web-based interface where users can select burgers from a predefined menu, customize toppings, and choose sides and drinks from dropdown menus. On the backend, we implemented a MySQL database to store and manage orders, including burger details, sides, and drinks.

The project uses Docker to manage the application environment and database, ensuring consistenc across different machines. We integrated Docker to run both the application and MySQL database, handling database migrations and ensuring data persistence. The frontend is built using EJS templates, providing dynamic HTML rendering. We also implemented error handling and ensured that the database is populated correctly by addressing file execution order and pathing issues in Docker. Lastly, GitHub was used for version control, with team members using feature branches to work independently and merge their work using pull requests.

vad ni gjort och hur ni tänkt för

Projektet i stort

We created a *Trello* to plan and structure our project to distribute the work equally.

Varje container

We created two containers for the project: one for the website and one for the database. Initially, we considered creating a third container for the kitchen view, but we realized it wasn't necessary. This was because the kitchen view only required a simple text output when a burger was ordered, so it could be integrated into the existing setup without needing a separate container.

Varje modul

We created separate modules for the burger order and kitchen view, placing them in two distinct file These modules are exported and then imported into the routes file. The reasoning behind this approach was to improve structure and organization, making it easier to clearly see which code belongs to each specific functionality.

4. Era erfarenheter om hur projektet gick att genomföra.

Vad gick bra?
Our collaboration throughout the project went smoothly. Each of us had an area of responsibility, bu we still worked together on all aspects of the project. However, each person ensured that their specific area was progressing and the necessary work was getting done.

Vad gick mindre bra?

We primarily focused on completing the project, which led to personal reflections being somewhat neglected. Our inconsistency with maintaining the project diary often filling it in every third day resulted in some information being forgotten.

Hur löste ni svårigheterna? Hade ni kunnat göra annorlunda?

Initially, we anticipated potential challenges with collaboration, as one of our group members moved to Japan and was in a completely different time zone. To work around this issue, we focused on making as much progress as possible during the hours he couldn't participate. At the end of our workday, we sent patch notes outlining what had been implemented and what still needed to be done. This approach allowed him to step in when needed, particularly when we got stuck or needed a fresh perspective on a problem. By the time we resumed work the following day, the issue was often resolved, which improved our efficiency. This process worked both ways—if he encountered difficulties, he would do the same for us.

Vad lyckades ni inte lösa? Varför inte?

We successfully implemented the required functions for the assignment. However, if this had been a program designed for a real client, additional functionality would have been necessary. Since the assignment only required basic, functioning software, we didn't focus on implementing more complex features. For example, the design and styling were kept generic and could have been more customized and unique had we allocated more time.

5. Era erfarenheter om att arbeta med containers.

We had some prior familiarity with working with containers, as they are part of other courses in our curriculum. However, despite this experience, we encountered several technical challenges during project. Nonetheless, as we worked through these issues, our understanding and knowledge of containers expanded significantly.

Vad gick bra?

Using Docker for containerization ensured consistency between development and testing environments. It simplified running the application and minimized the number of packages that each team member needed to download, as the Dockerfiles handled all the necessary dependencies. Th docker-compose.yml file streamlined the database setup by managing login details, ensuring the application ran consistently across different machines and reducing the "it works on my machine" problem.

As mentioned earlier, the docker-compose.yml file made setting up BurgerHub straightforward by allowing us to define and manage multiple services (e.g., databases, backend) with a single comma Additionally, debugging within the container was easily managed through the Docker Desktop application.

Vad gick mindre bra?

We encountered some platform-specific issues that were challenging to solve, as identifying the roo cause was difficult. The database worked on Amir’s and Theodore’s terminals but not on Alem’s an Tuva’s. After extensive research, we discovered that tools like Git Bash can behave differently on L and Windows, particularly in terms of file permissions. This realization helped us trace the issue ba to the fact that the working terminals were Ubuntu-based.

Another problem we faced was Docker ignoring the project's my.cnf file because it was writable. Th behavior is a security measure designed to prevent potential vulnerabilities.