Software Architecture and Communication

System Integration

Ola 1

Kristofer Pedersen
cph-kp278

Mads Knudsen
cph-mk682

Michael Frederiksen
cph-mf315

Søren Merved
cph-sm428

September 8, 2024

Contents

1 Brainstorm

Before starting our technology stack and development stack, we decided to set up a business case from which we could structure our stacks around. Instead of creating a new case from scratch we instead choose to reuse a previous business case from an exam project from our previous education: The cupcake business.

The idea revolves around a business wanting to sell cupcakes to their customers through a website. The user (customer) would go to the website where they could customize their cupcakes and add them to a basket. When the user decides to purchase they move to a dedicated page. Any orders created by a user using an email would then be viewable when the user has logged into the website and navigated to a user order page.

From the cupcake business case we can expand on the services offered using different microservices. We can add review functionality by using microservices offered by businesses that revolve around the products being reviewed.

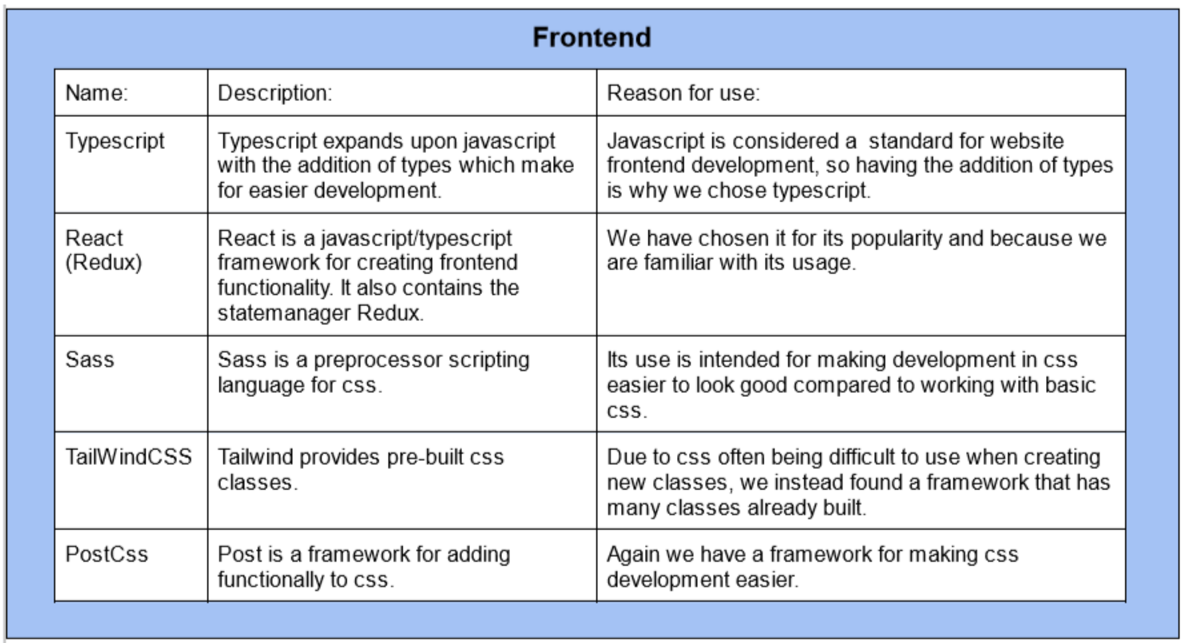
Using our business case we have created multiple different user stories to help convey the intended functionality of the website from frontend through the api to the backend.

1.1 User stories

- As a user, I want to be able to login, so that I can keep track of my information such as credentials and previous purchases.
- As a user, I want to be able to browse through a possible selection of cupcakes and potential toppings.
- As a user, I want to be able to see images of available cupcakes, so that I can get a visual representation of what I am purchasing.
- As a user, I want to be able to add cupcakes to a basket which holds my current selection.
- As a user, I want to be able to view and edit my basket.
- As a user, I want to be able to confirm my basket, so I can complete the purchase of my selected cupcakes.
- As a user, I would like to track my delivery.
- As a user, I want to be able to reach out to a support service if I need help.

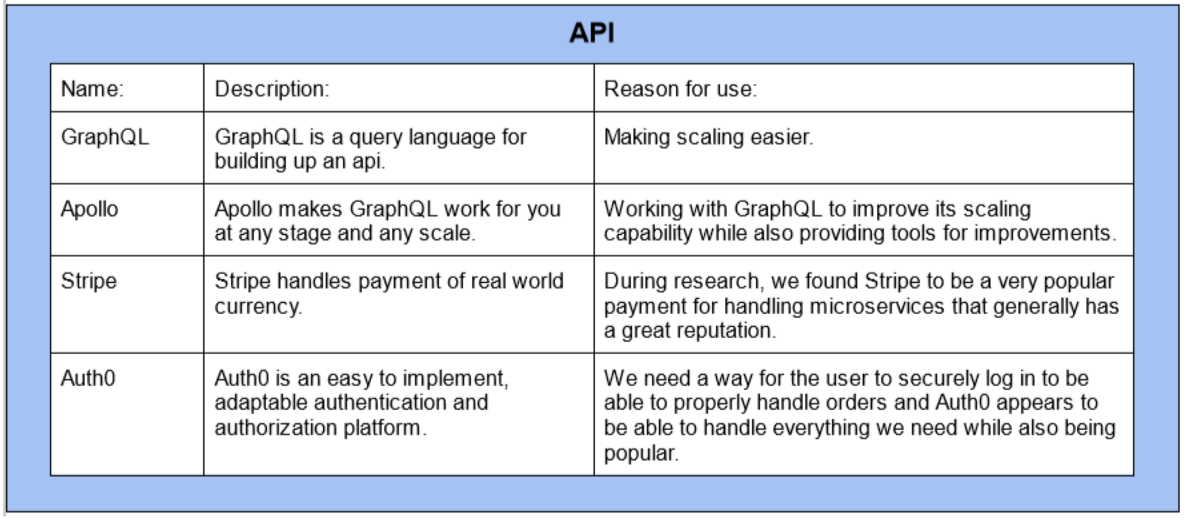
2 Technology Stack

When building a tech stack we can separate the different technologies used into three different categories based on where the technologies are used. First we have the frontend where the user will interact with the website. Then we have the backend where we handle the data of the business and finally we have the api that ties them together.



Frontend

Name:	Description:	Reason for use:
Typescript	Typescript expands upon javascript with the addition of types which make for easier development.	Javascript is considered a standard for website frontend development, so having the addition of types is why we chose typescript.
React (Redux)	React is a javascript/typescript framework for creating frontend functionality. It also contains the statemanager Redux.	We have chosen it for its popularity and because we are familiar with its usage.
Sass	Sass is a preprocessor scripting language for css.	Its use is intended for making development in css easier to look good compared to working with basic css.
TailWindCSS	Tailwind provides pre-built css classes.	Due to css often being difficult to use when creating new classes, we instead found a framework that has many classes already built.
PostCss	Post is a framework for adding functionally to css.	Again we have a framework for making css development easier.

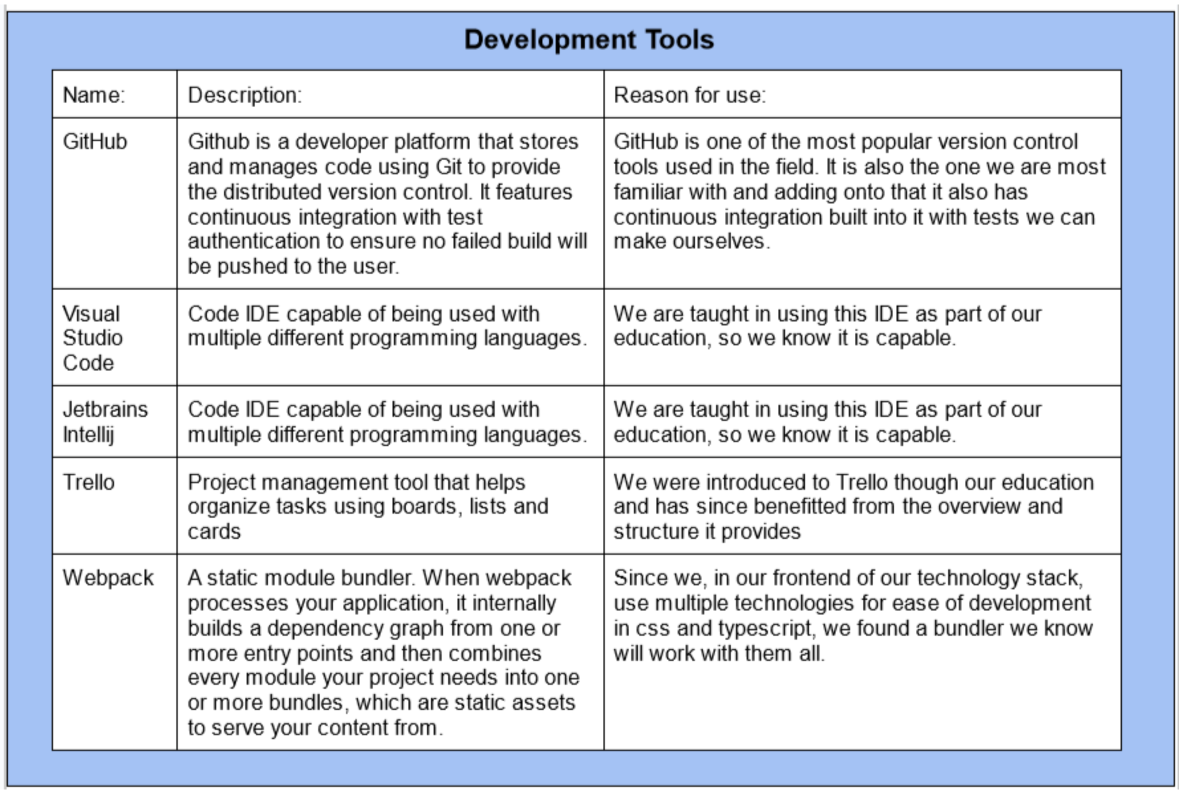


API

Name:	Description:	Reason for use:
GraphQL	GraphQL is a query language for building up an api.	Making scaling easier.
Apollo	Apollo makes GraphQL work for you at any stage and any scale.	Working with GraphQL to improve its scaling capability while also providing tools for improvements.
Stripe	Stripe handles payment of real world currency.	During research, we found Stripe to be a very popular payment for handling microservices that generally has a great reputation.
Auth0	Auth0 is an easy to implement, adaptable authentication and authorization platform.	We need a way for the user to securely log in to be able to properly handle orders and Auth0 appears to be able to handle everything we need while also being popular.

Backend

Name:	Description:	Reason for use:
NestJS	NestJS is a Node.js framework for building efficient and scalable server side applications.	We have chosen to use NestJs as it provides a lot of structure through a modular architecture. It also has good integration with both Typescript and TypeORM.
Node.js	Node.js is a Javascript runtime environment that enables running Javascript on the server side.	Using Node.js allows us to use Typescript for both frontend and backend, which simplifies development. Node.js also has a wide range of libraries and frameworks available through the node package manager (npm).
Nginx	Nginx provides a web server, and can also serve as a reverse proxy and HTTP cache.	We use Nginx as a reverse proxy and for load balancing to route requests to different backend servers, which improves availability and scalability.
TypeORM	TypeORM is an ORM framework for JavaScript and Typescript, which uses object-oriented programming principles to help manage database operations.	TypeORM allows us to define our database schema using Typescript classes, which helps work with relational databases in an object-oriented way.
Docker	Docker allows packaging applications into containers, providing consistency across different environments and making it easy to deploy	Docker ensures every team member has the same environment, removing the risk of version based errors and configuration inconsistencies
Digital Ocean	DigitalOcean is a cloud based platform that provides virtual servers and services for deploying/scaling applications.	We have chosen to use DigitalOcean since it allows us to easily deploy our application.
MySql	MySQL is a relational database that allows users to store, manage and retrieve data. It's commonly used in web applications and services due to its reliability and scalability.	A relational database like MySQL is ideal for an e-commerce website due to its reliable performance and ability to handle complex queries for managing data. Its scalability ensures that the database can grow with the business and handle increasing traffic and data volumes efficiently.
Internal microservices	Considering we would be making a website for a business that most likely would include other departments like the bakery, accounting, storage and so on, they would most likely have some service set up to handle each area which we wouldn't be making/developing.	Any department within the business which has their own systems set up to handle their part of the business, we would connect to them via an api they have set up. This also separates the responsibility to each department and makes the development of the web store easier to handle.



Development Tools

Name:	Description:	Reason for use:
GitHub	Github is a developer platform that stores and manages code using Git to provide the distributed version control. It features continuous integration with test authentication to ensure no failed build will be pushed to the user.	GitHub is one of the most popular version control tools used in the field. It is also the one we are most familiar with and adding onto that it also has continuous integration built into it with tests we can make ourselves.
Visual Studio Code	Code IDE capable of being used with multiple different programming languages.	We are taught in using this IDE as part of our education, so we know it is capable.
Jetbrains Intellij	Code IDE capable of being used with multiple different programming languages.	We are taught in using this IDE as part of our education, so we know it is capable.
Trello	Project management tool that helps organize tasks using boards, lists and cards	We were introduced to Trello though our education and has since benefitted from the overview and structure it provides
Webpack	A static module bundler. When webpack processes your application, it internally builds a dependency graph from one or more entry points and then combines every module your project needs into one or more bundles, which are static assets to serve your content from.	Since we, in our frontend of our technology stack, use multiple technologies for ease of development in css and typescript, we found a bundler we know will work with them all.

3 Enterprise integration exploration

3.1 What it is

Enterprise integration is the overall term for the use of one or more integration approaches, that includes API management, application integration and messaging to be able to leverage enterprise services and assets, as to expose them as APIs or to connect them as services. This enables organizations to seamlessly integrate, unify and standardize their core business capabilities across diverse IT environments, which specifically can enable you to easily do the following four:

- Access and expose application functions via APIs.
- Discover valuable services, applications and data.
- Monitor application lifecycles and governance.
- Connect multiple enterprise services.

We found that enterprise integration is important because it's the key to enhance internal processes and business activities and the conceptualization, implementation and the distribution of critical applications and/or services. Companies can improve by sharing important information, simplifying their processes and maximizing their opportunities, while also improving their operational scalability and increasing their reach and revenue.

For enterprise integration we found that it can be boiled down to five different key elements, that being: APIs, application integration, messaging, events and data.

APIs or application programming interfaces are one of the most well known programming elements that are part of enterprise integration. They are for example between the application and the server, and they enable different companies to share data and functionality of their individual applications with third-party developers, business partners and their own internal departments. APIs are increasingly being used to access and expose real-time data throughout the wide world, and can be extended to as more sources, such as data published as events.

Application integration is the part about enterprise integration that is about enablement of individual applications, where each application is designed for a specific purpose, while working collaboratively. By making the data from individual applications easier to share and to combine their workflows and processes, means organizations can see benefits from integrations that can modernize their infrastructures without having to rework everything.

Messaging means to help provide resilience and performance to IT environments that spans the cloud and also on-premises systems. Messaging must be able to cross network boundaries to be able to provide a reliable delivery while also preserving network-wide message integrity, data protection and regulation compliance via implementation of security-rich functions.

Events refers to records of action or a form of change. It refers to when one application or a service performs an action or undergoes some form of change relative to the functionality of another application or service. When one application publishes an event, other applications or services can detect the event publication, and from there they can process the event, perform one or more reciprocal action/change or simply ignore the event. In programming terms, this can be compared directly to the listener pattern.

Data, specifically real-world operational data, that enables continuous improvement/integration, which is known as CI, of enterprise architecture. The data is also used to assess the criticality and usage of integrations and to determine their target state. When the data is analyzed, it will reveal recommended target integration patterns, like service-oriented architecture or SOA, or event-driven or message-driven, which consolidate the possibilities and other inputs that can help define the target integration state.

4 Its effect

Enterprise integration significantly impacts an organization's efficiency, scalability, and ability to adapt to changing market demands. By enabling seamless communication between disparate systems, it facilitates sharing critical information across departments and external partners, ensuring real-time access to valuable data. This integration enhances decision-making and drives innovation by allowing teams to work with accurate, up-to-date insights.

Moreover, enterprise integration helps in simplifying IT processes, streamlining workflows, and reducing the complexity of managing multiple applications. It eliminates the need for siloed systems, reducing redundancy and manual effort, which improves operational efficiency.

Lastly, it enables organizations to maximize opportunities by creating a flexible infrastructure that can quickly integrate new technologies, services, or applications. This agility allows businesses to respond faster to market trends, customer needs, and competitive pressures, thus boosting their operational scalability, expanding their reach, and increasing revenue.

5 Examples of integration patterns

5.1 Examples of common patterns

Data-Centric Integration is a strategy that revolves around establishing a single source of truth for the data within an organization. This approach emphasizes accuracy, consistency and the reliability of the data across various applications and systems. Some of the common patterns that are associated with data-centric integration are: ETL (Extract, Transform, Load), File Transfer, Shared Database, Data Replication and Aggregation. Some of the benefits regarding the use of data-centric integration patterns are: Consistency, Efficiency and Accuracy

Event-Driven Integration is a strategy that focuses on the enabling of real-time responsiveness and agility in system architectures through the use of loosely coupled communication. This approach is centered around the idea that systems can react to events as they occur which allows for more dynamic and flexible interactions between the different components in a system. Some common patterns for this strategy are: Message-Driven Communication and Event-Driven Communication. The benefits for this strategy are: Real-time responsiveness, Scalability and Flexibility.

Application-Centric Integration is an approach that focuses primarily on promoting modularity, reusability, and maintainability in the application using well-defined interfaces and APIs. This strategy emphasizes building an application as a set of loosely coupled, modular components that can communicate with each other seamlessly. Some of the common patterns with this strategy are: Facade, Adapter and Content-Based Routing.

6 Examples of real world usage

MNG Kargo, which is a leading transport and delivery company based in Turkey, has seen an increase in their business during the recent e-commerce boom. And to meet the rising demand they were facing, they chose to do the following:

- Used IBM's secure gateway to launch an API developer portal that can establish connections with e-commerce providers, allowing seamless data flow from sale to delivery.
- Used IBM API management to automate API-based connections with partners.
- Added a bit of innovation. They hosted a hackathon where third-party developers proposed solutions (and code) for service enhancements.

Another company that uses enterprise integration is Helsinki Regional Transport Authority or HSL, who are responsible for public transportation for up to 1.5 million people in the Helsinki metro region. When they found they needed to update their internal ticketing system they choose to do the following:

- Moved from traditional virtual machines to microservices with Red Hat OpenShift.
- Used IBM Cloud Pak for Integration to connect their applications and data directly to various systems throughout their company.
- Completed their system migration — during a global pandemic — without any unplanned outages.