WIE FUNKTIONIERT DAS INTERNET

Fortgeschrittene Informatik

Ziele der Vorlesung

- **eigenverantwortliche** Einarbeitung und Bearbeitung im Bereich Webentwicklung
- Konzepte und Technologien der Internet-Kommunikation
- verteilten Systeme verstehen
- Umsetzung von Web-Anwendungen
- Web-Architekturen

Verteilte Systeme

Verteilte Systeme

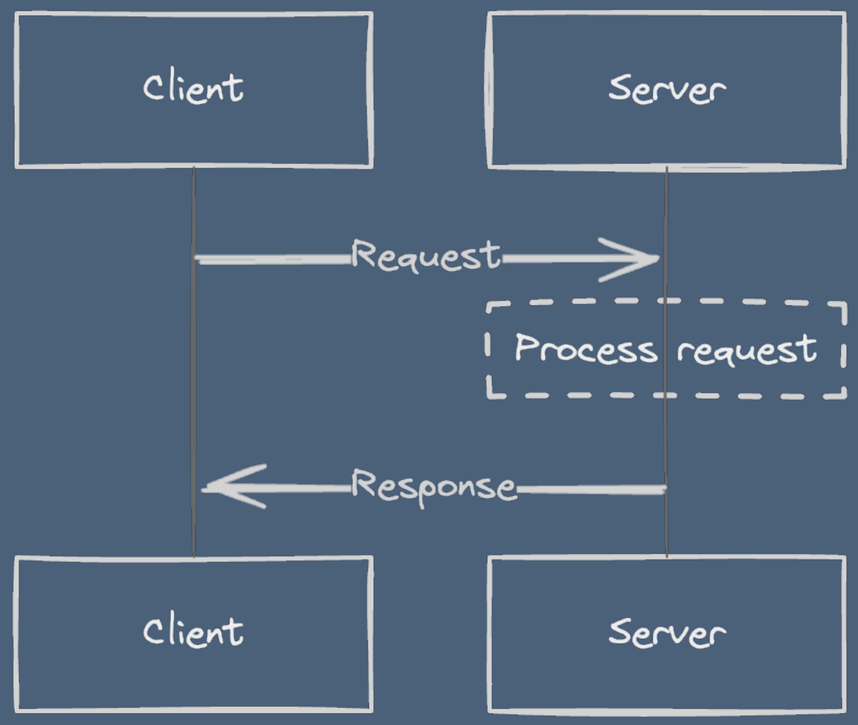
Das Internet ist das bekannteste verteilte System der Welt

- Galli: “A distributed system is a collection of heterogeneous computers and processes connected via a network. This collection works closely together to accomplish a common goal.”
- Tanenbaum: “A distributed system is a collection of independent computers that appears to its users as a single coherent system.”
- Ein verteiltes System besteht aus einer Ansammlung von Computerprogrammen, die Rechenleistung über mehrere unabhängige Rechenknoten verteilen, um ein gemeinsames Ziel zu erreichen. Der Einsatz verteilter Systeme zielt darauf ab, Engpässe oder zentrale Schwachstellen im System zu umgehen.

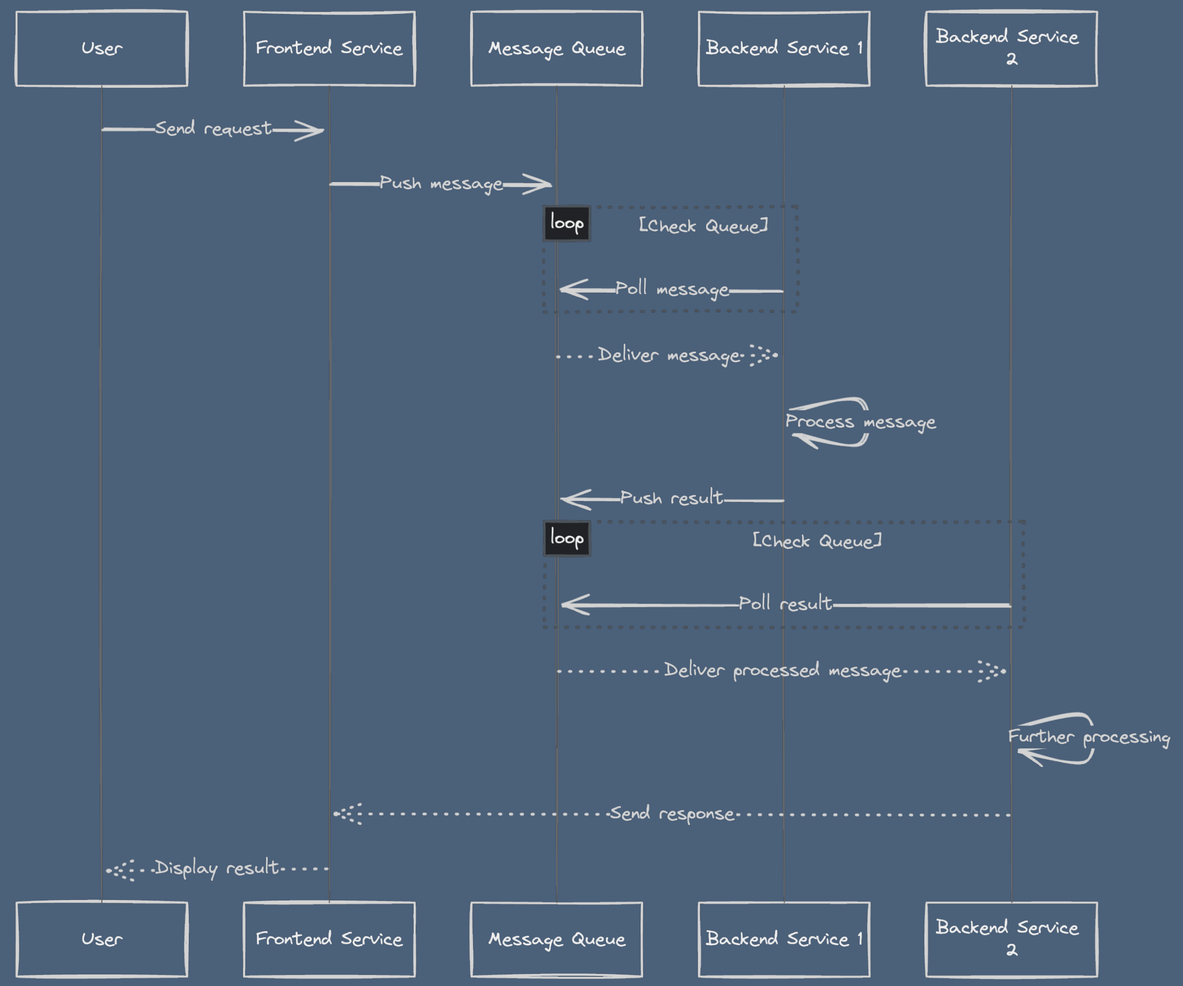
Verteilte Systeme

Kommunikation

- Synchrone Kommunikation (z.B. REST)

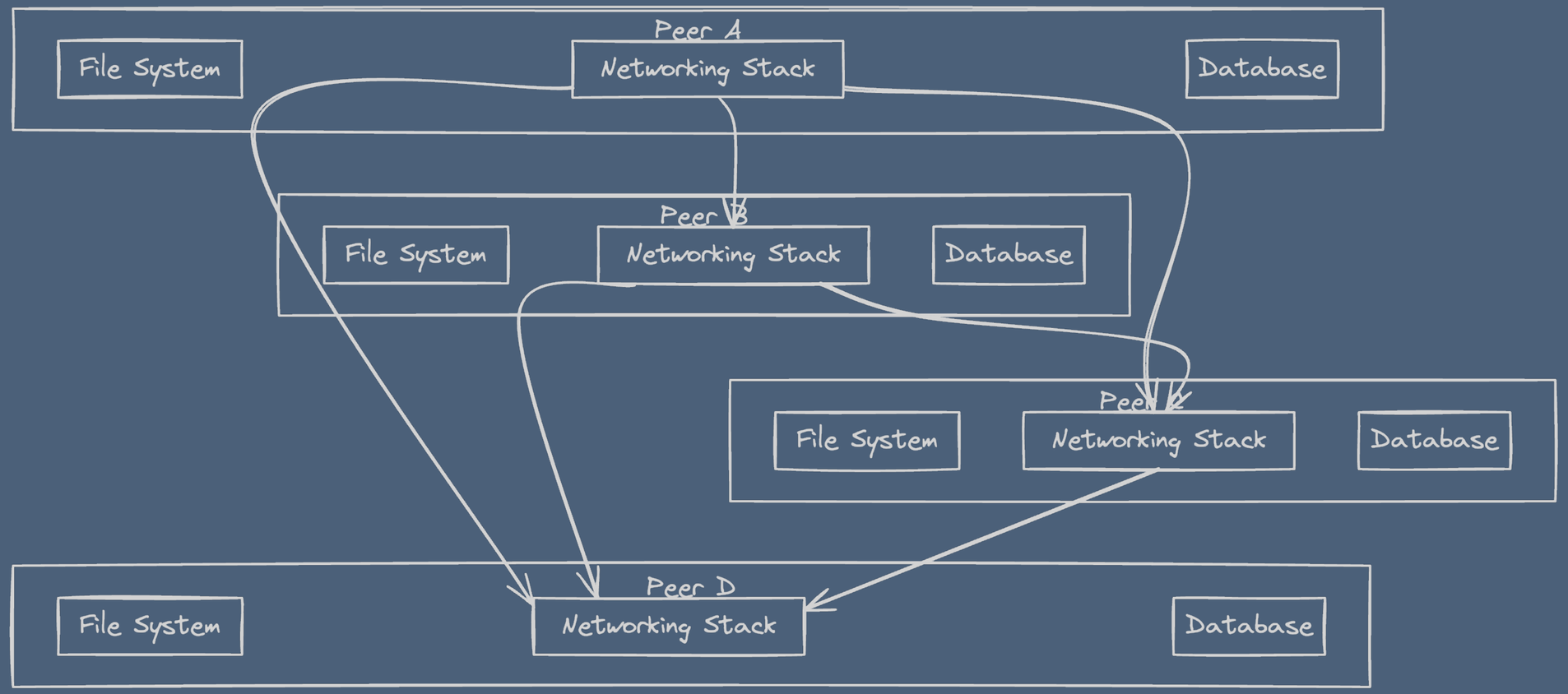


- Asynchrone Kommunikation (Backend Prozessing)



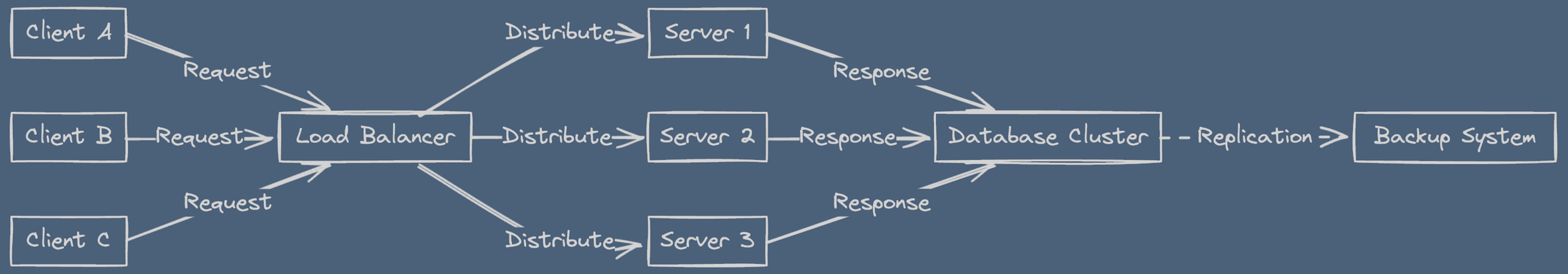
Architektur Verteilter Systeme

Systemarchitektur - Peer to Peer



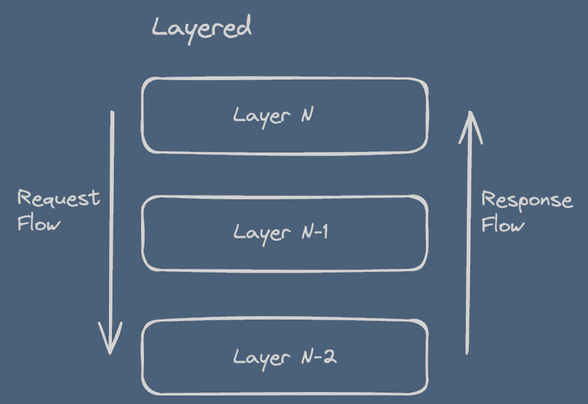
Architektur Verteilter Systeme

Systemarchitektur - Netzwerk

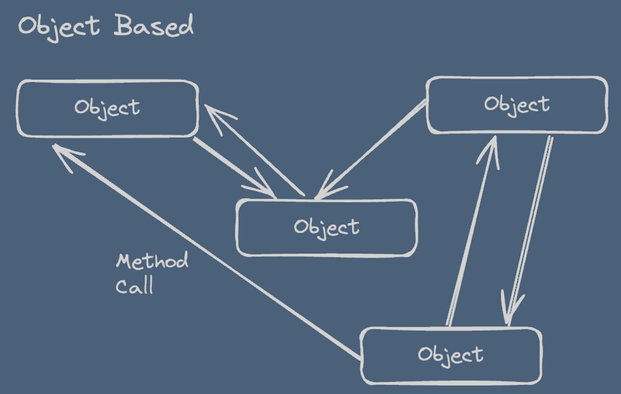


Architektur Verteilter Systeme

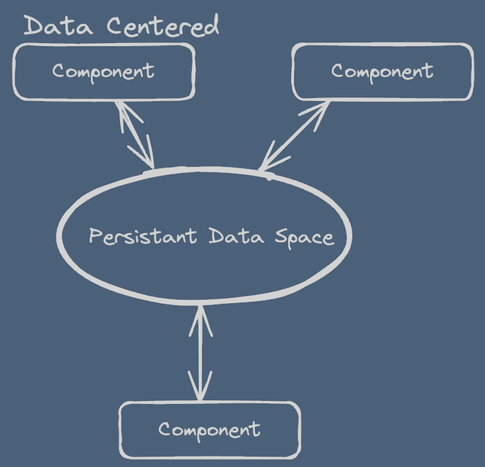
Softwarearchitektur



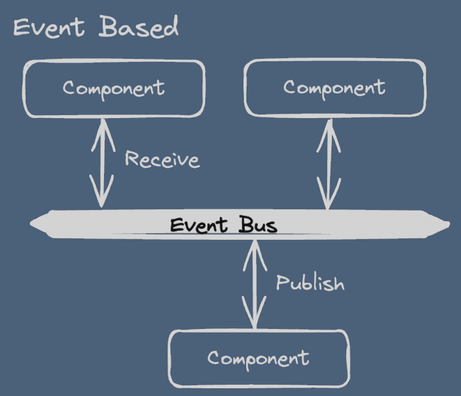
- Frontend, Server, Database



- Webservices and Rest API



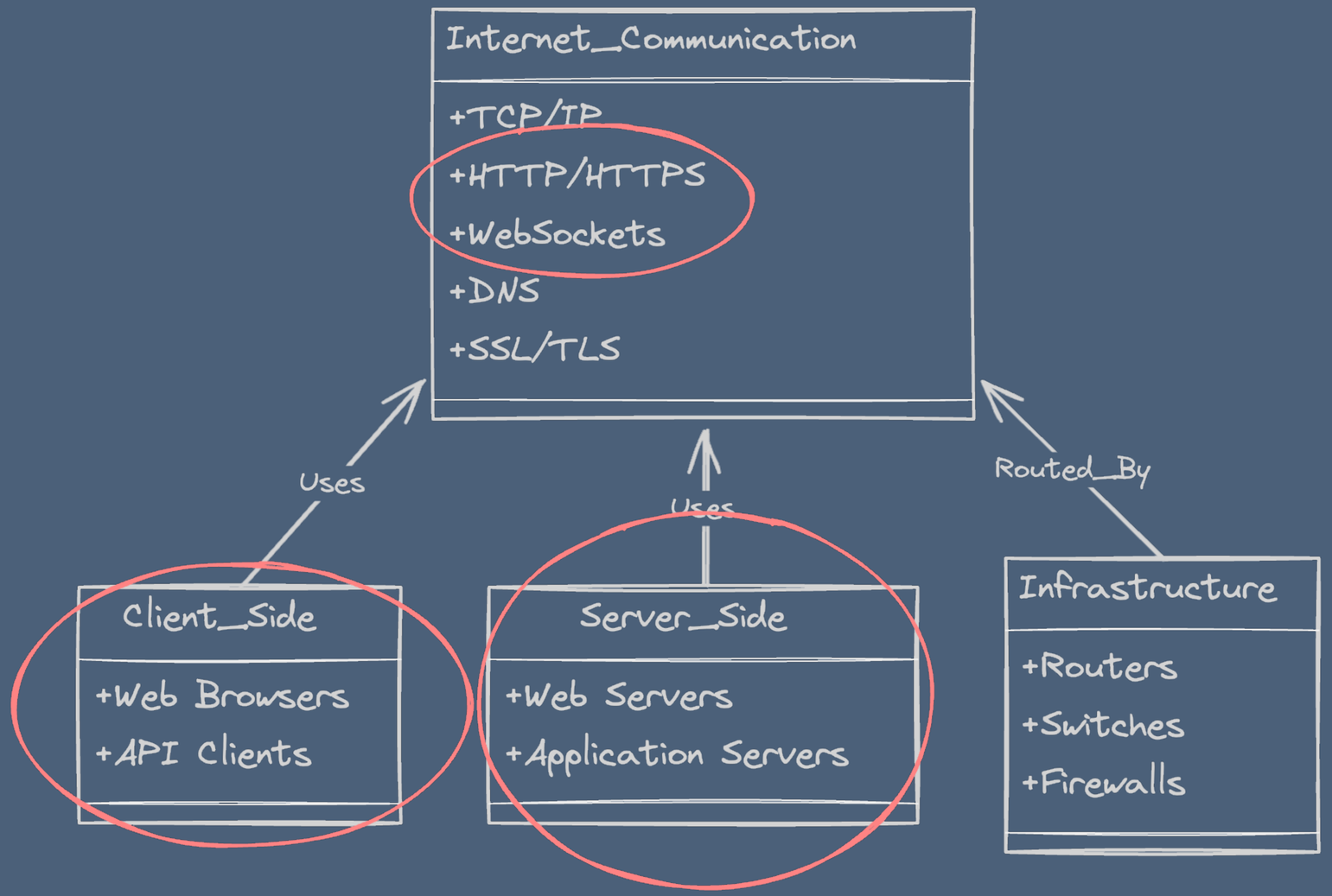
- Databases, CRM



- Apache Kafka

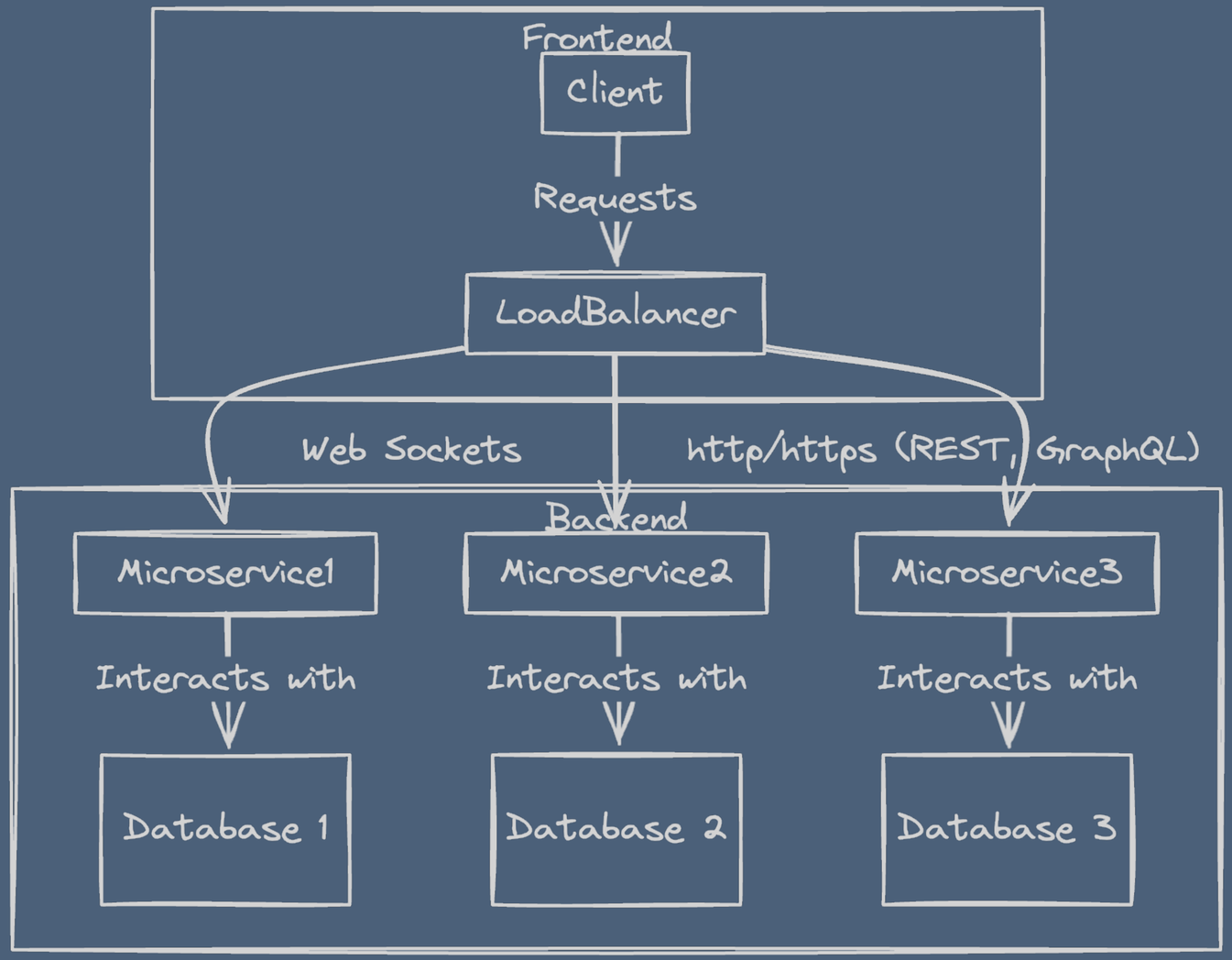
So funktioniert das Internet

So funktioniert das Internet



So sieht ein Web-Entwickler das Internet

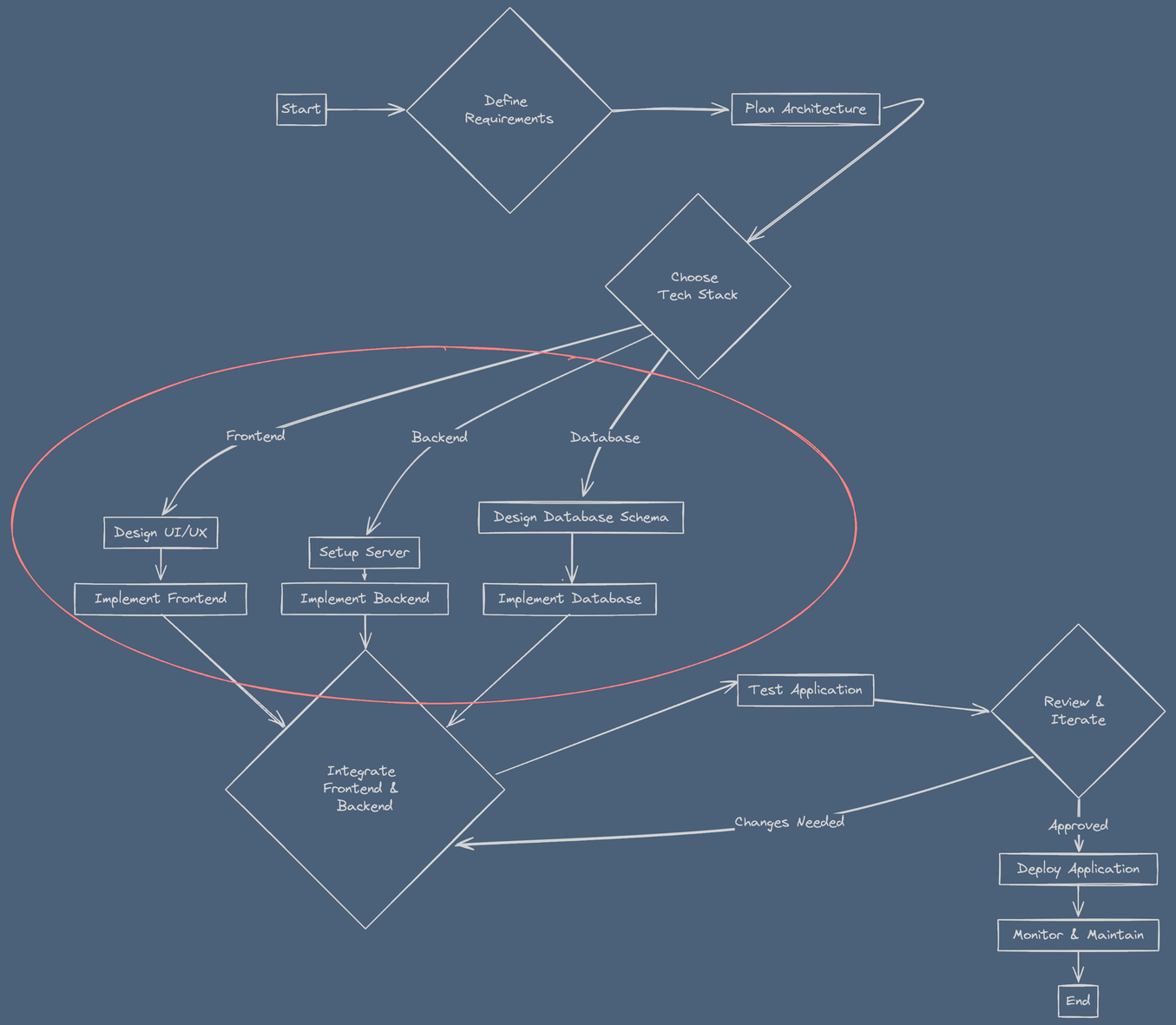
FULLSTACK Perspektive



Begriffe, über die Ihr stolpern werdet

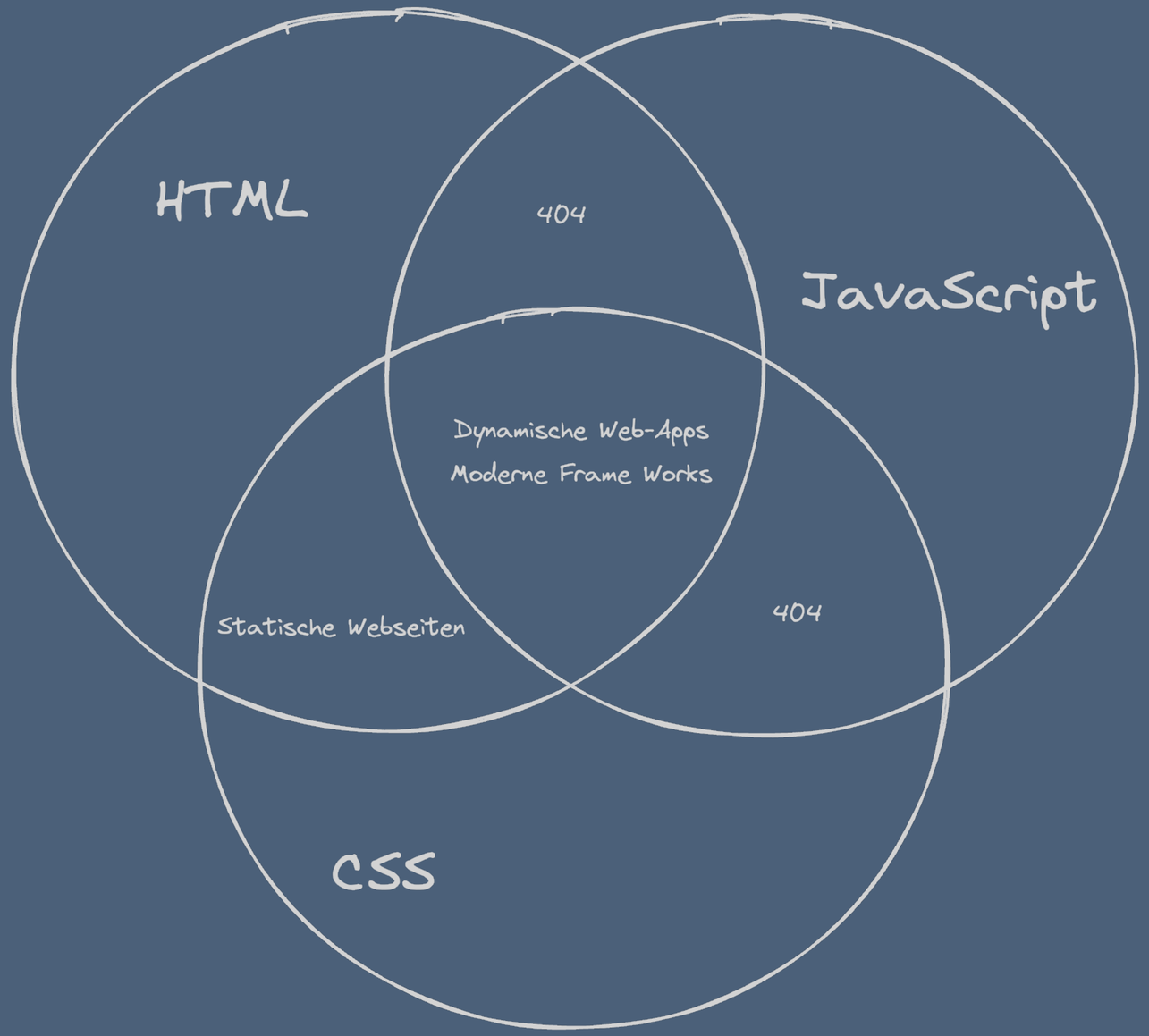
- Page Builder
- SPA
  - Client Side Rendering
  - Server Side Rendering
- PWA
- Load Balancer
- Cloud Solutions
- (Headless)CMS
- CDN

Der perfekte Entwicklungsprozess



Das Frontend

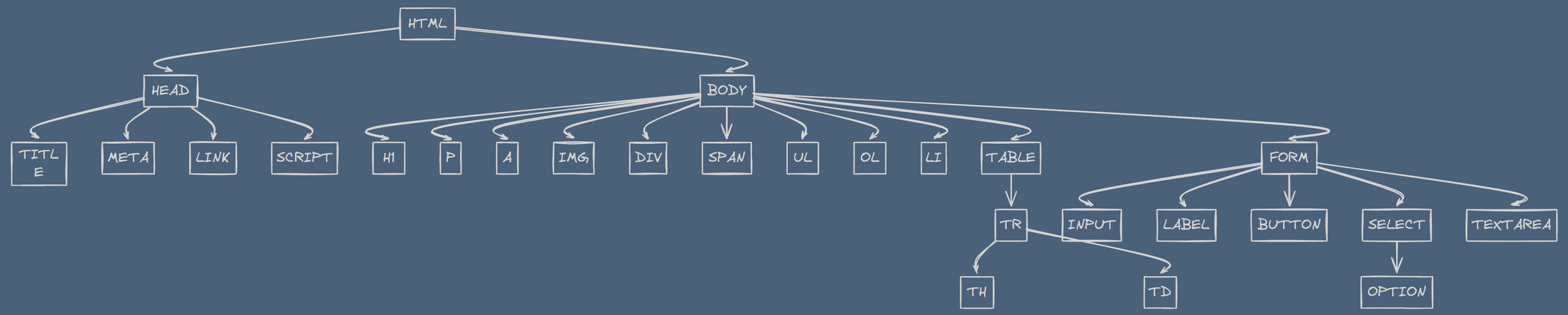
Das Frontend



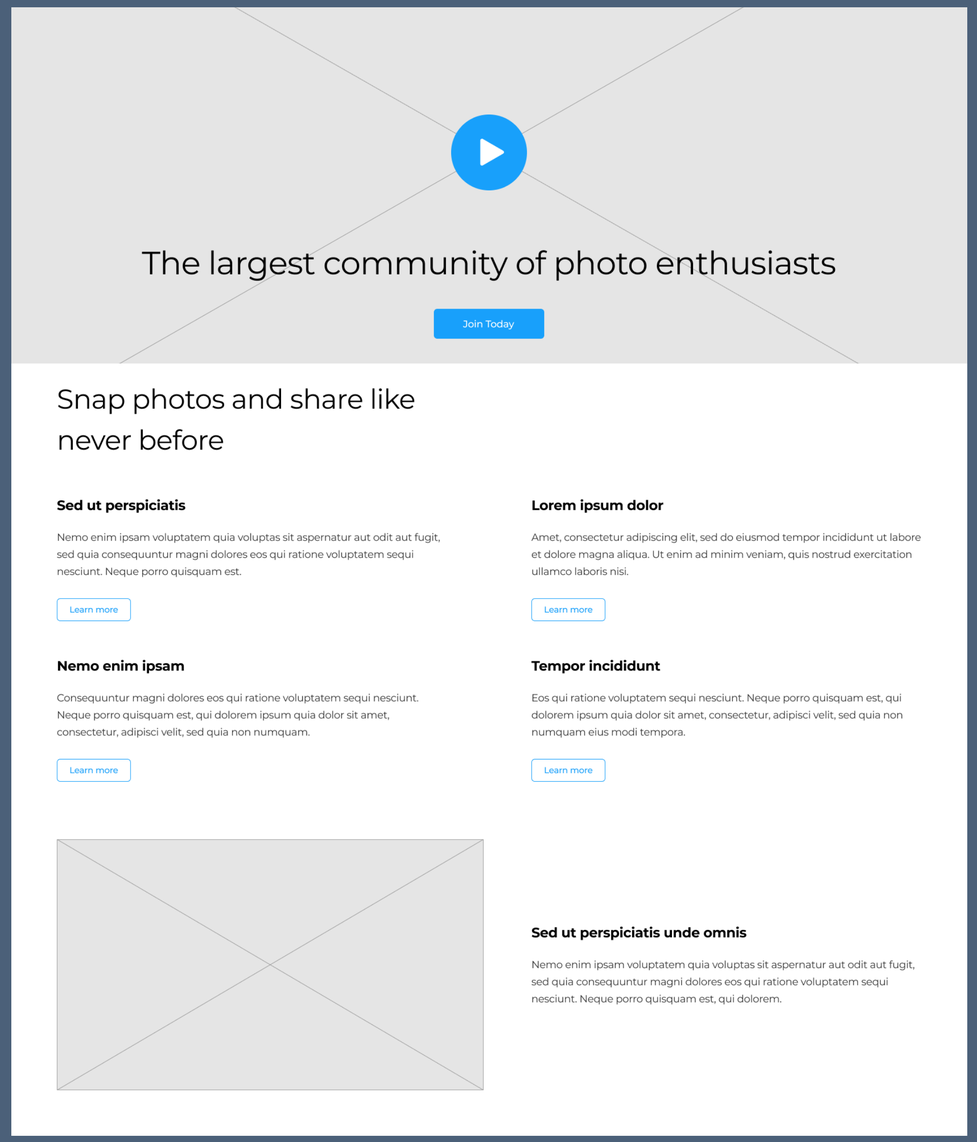
HTML - Hyper Text Markup Language

https://www.w3schools.com/tags/default.asp

- Entstand in den 80/90er Jahren für CERN
- Beschreibt den Inhalt eines Dokumentes
- Basiert auf Tags und Baumstruktur



HTML-Beispiel



HTML Project

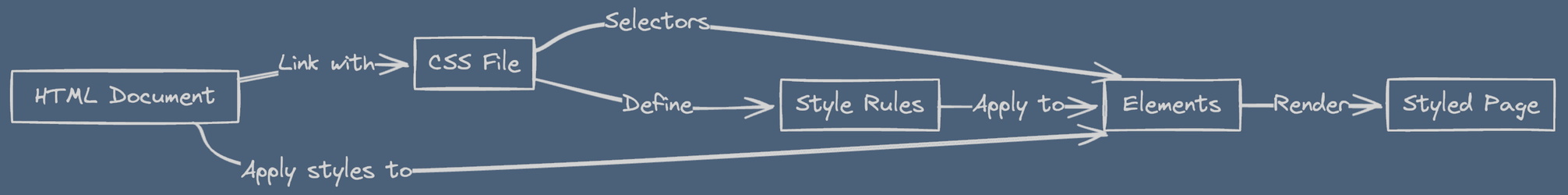
Restaurant Website

- Erstellt die HTML Struktur einer Website für ein Restaurant, die folgende Informationen enthält:
  - Name des Restaurant
  - Adresse des Restaurant
  - Öffnungszeiten des Restaurant
  - Ein Bild des Restaurant
  - Die Speisekarte

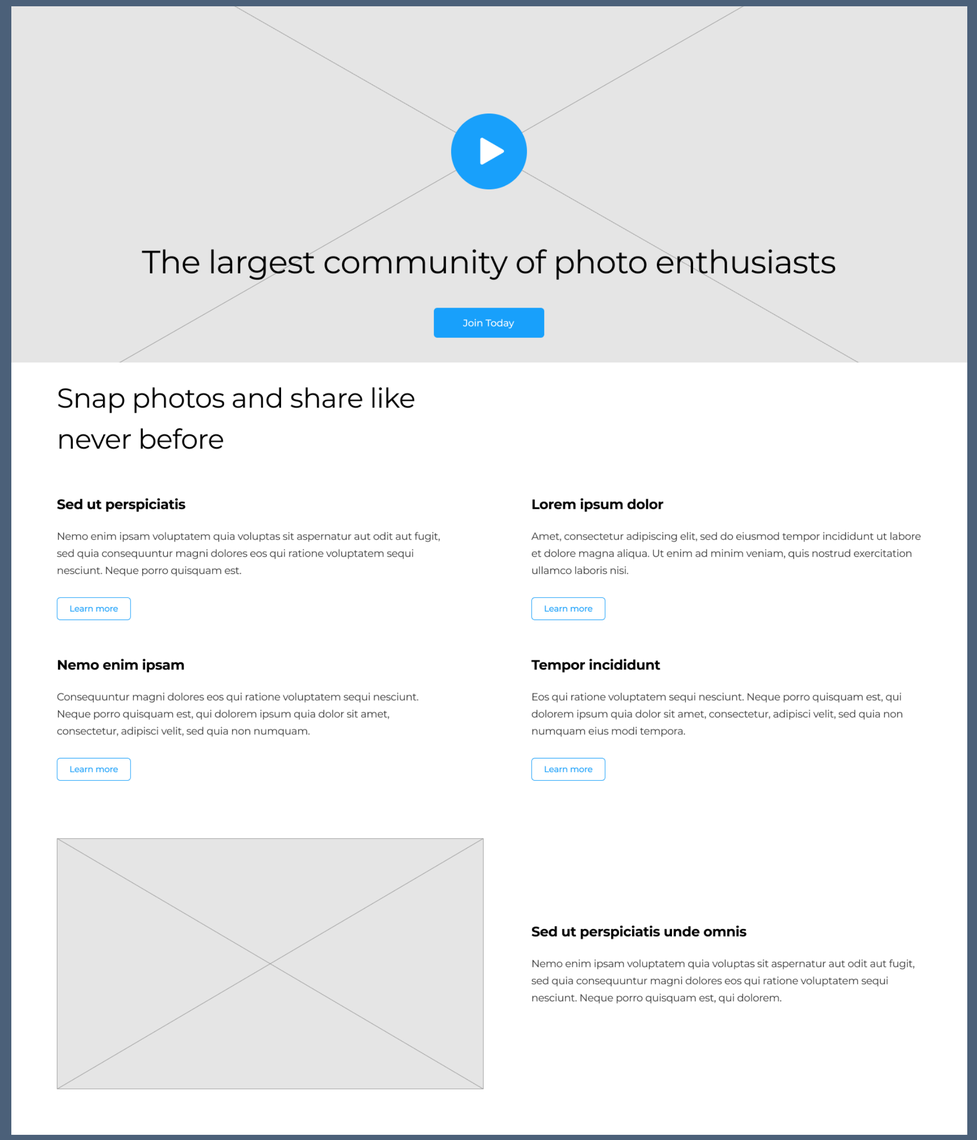
CSS - Cascading Style Sheets

https://wwww3schools.com/cssref/index.php
https://wwww3schools.com/cssref/css_selectors.php

- Entstand Mitte der 90er Jahre
- Beschreibt die Optik von Dokumenten
- Basiert auf Selectoren, Pseudo-classes, Combinators und Rule-Blöcken



CSS-Beispiel



CSS Project

Restaurant Website

- Strukturiert die Webseite des Restaurant nach euren wünschen und verwendet hierfür mindestens:
  - einmal FlexBox
  - einmal Grid
  - einen Class-Selector
  - einen ID-Selector
  - einen Element-Selector

Java Script

- Just-in-time compiled Script Sprache
- Initial aus dem Jahr 1995
- Die „Client Programmiersprache im Internet“
- Wird verwendet um die Inhalte im Client zu dynamisieren
- Basis einer Vielzahl von Frameworks die zur Entwicklung von Web-, Nativen- und Cloud-Apps verwendet werden

JavaScript-Beispiel

DOM Manipulation

Java Script Project

Restaurant Website

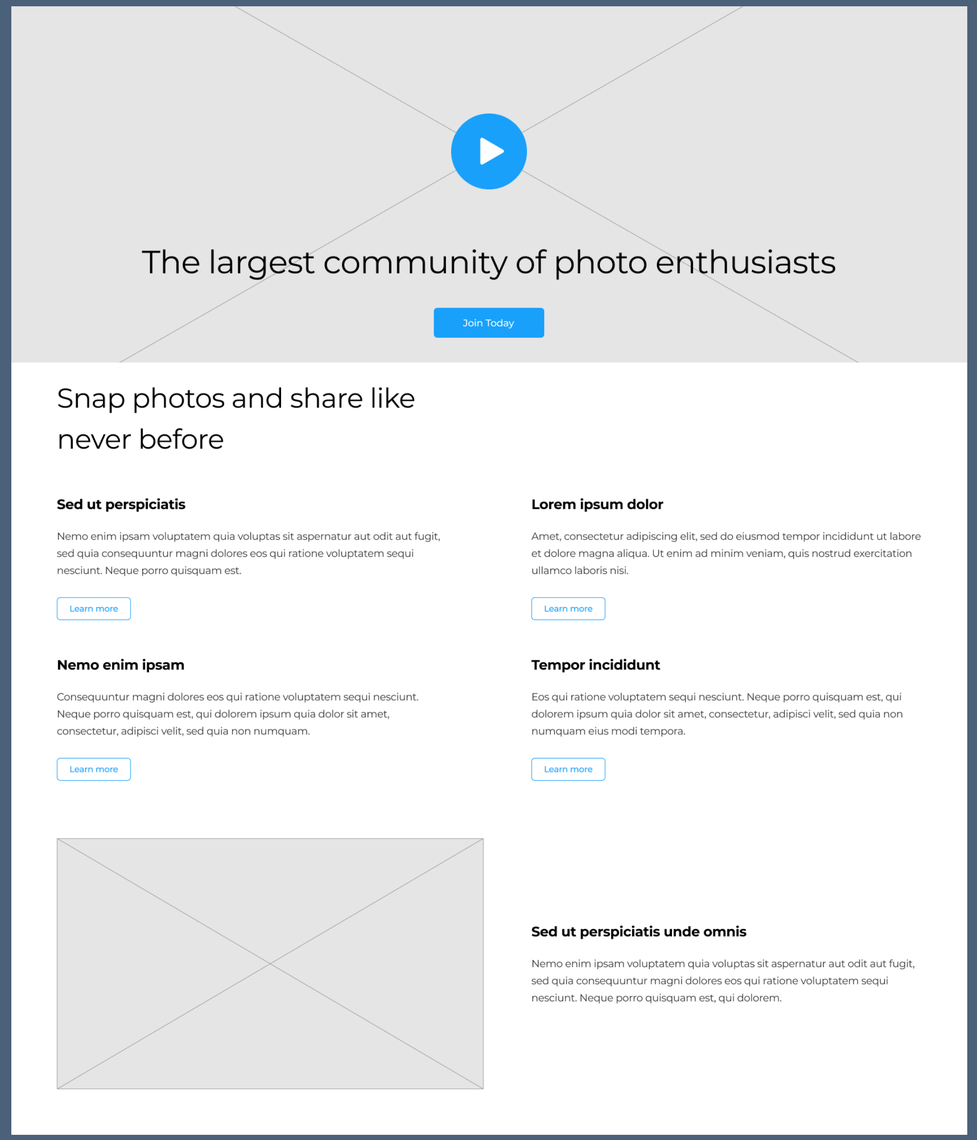
- Integriert folgende Elemente in die Restaurant Website:
  - Ein Reservierungsformular mit einem Senden-Button
    - Die Inhalte des Formulars werden als Object nach dem Clicken des Buttons in die Console gesendet
    - Nach Klicken des Buttons verschwindet das Formular und der Kunde bekommt den Schriftzug „Vielen Dank für Ihre Buchung zu sehen“

Moderne Webframeworks

React

- Open Source JavaScript Bibliothek von Meta entwickelt
- Grundlage diverser Frameworks z.B. Next oder Gatsby
- Initial 2013 vorgestellt, inzwischen in Version 18
- Basiert auf Komponenten, virtuellem DOM und Routing
- Verwendet als Sprache JSX/TSX

React-Beispiel



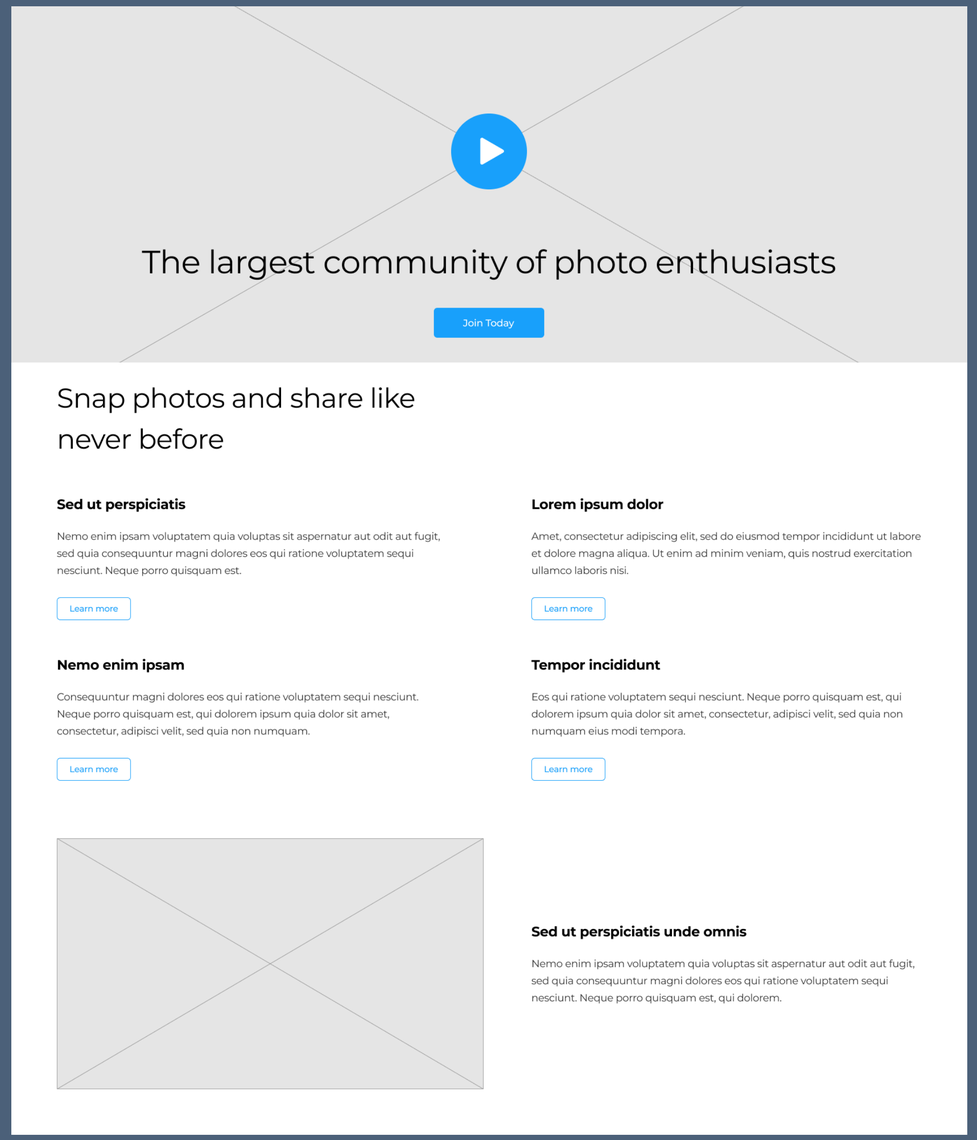
React Project

Restaurantwebsite

Next JS

- Open Source Fullstack Framework basierend auf React
- Initial aus dem Jahr 2015
- Aktuelle Version: 14
- Static-Side generation, Server Side Rendering, Client-Side rendering und API-Implementierung in einem Projekt möglich

NextJS-Beispiel



NextJS Project

Restaurantwebsite

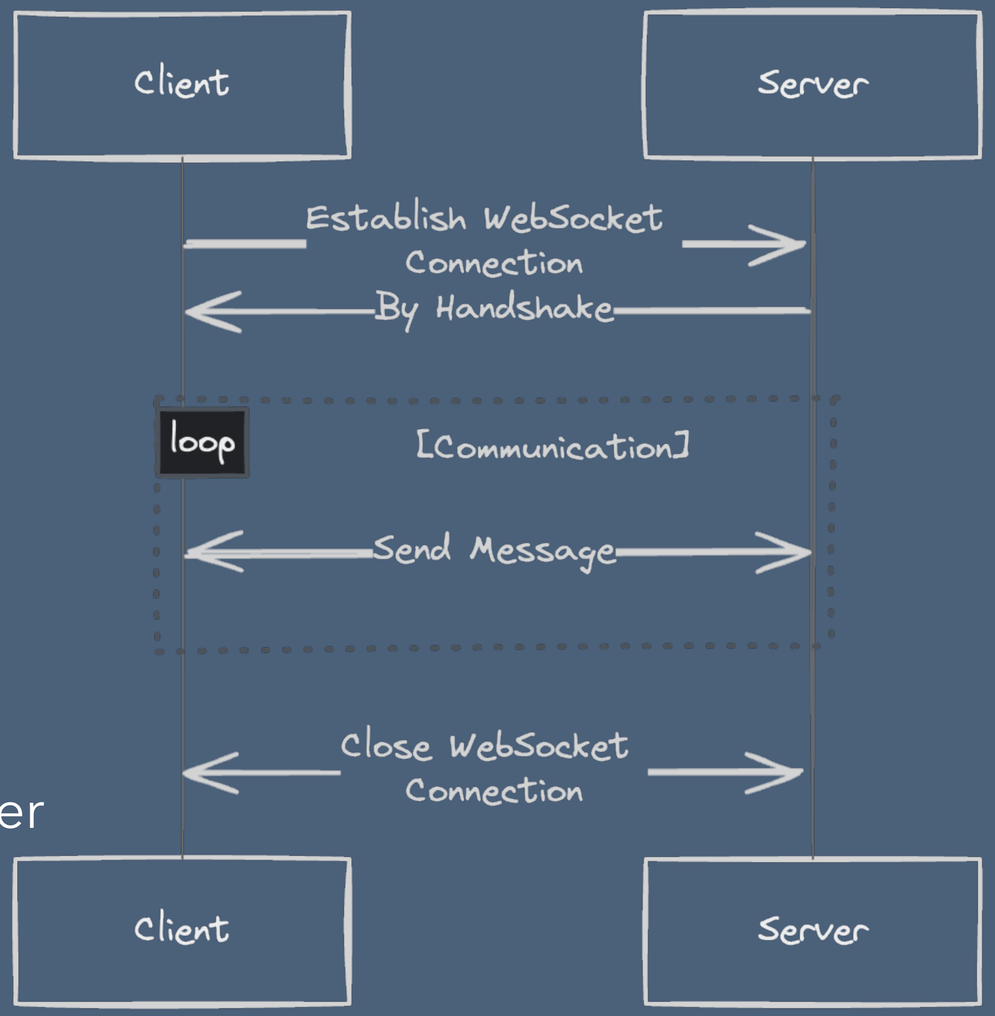
Das Backend

Das Backend

- APIs
  - REST API
    - GET, POST, PUT, DELETE
  - GraphQL
- Data-Processing
- Automatisierung

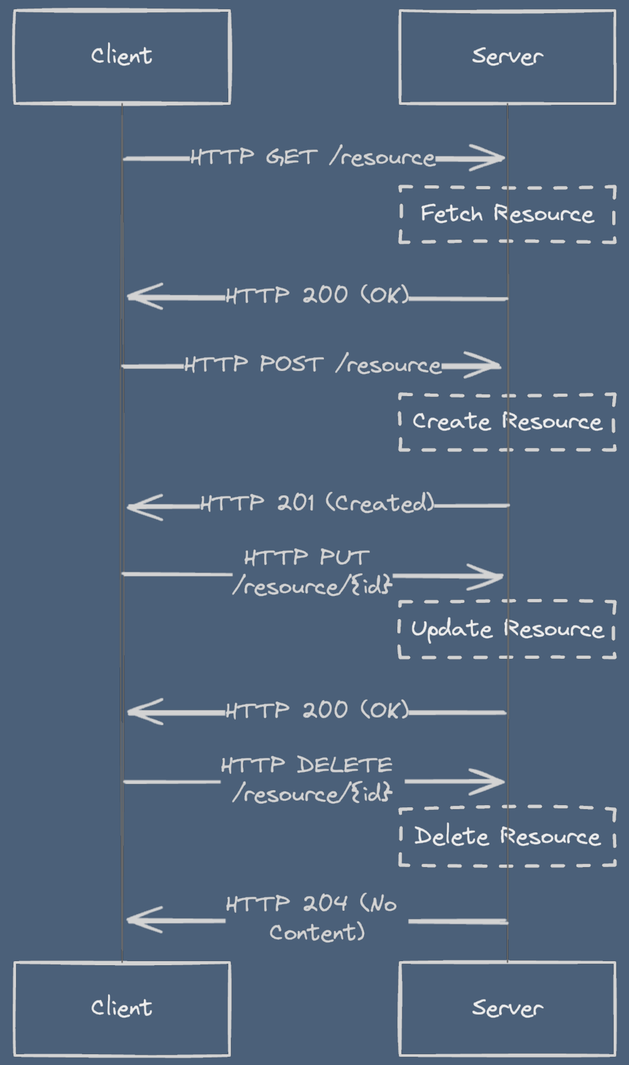
WebSockets

- Kommunikationsprotokoll
- Basieren auf TCP
- vollduplex Verbindung zwischen Client & Server
- Echtzeitkommunikation
- Einsatzbereiche:
  - Verkaufsplattformen, Chats, Social Media, Live Ticker



REST API - Representational State Transfer

- Architekturvorgaben, aber kein Protokoll oder Standard
  - Zustandslos
  - Einheitliche Schnittstelle
  - Entkopplung von Client und Server
  - Cachefähig
  - Mehrschichtigkeit
  - Frameworks in JavaScript: z.B. NestJS oder ExpressJS
  - https://rapidapi.com/



Der Code ist unter
/tables/route.js
Und
/components/Btn.js

Projekt API

Weather for Location

- https://open-meteo.com/en/docs
- https://developers.google.com/maps/documentation/geocoding?hl=de

FrontEnd

Wie transformiere ich das Design in Code

Welche Inhalte sind statisch

Welche Inhalte sind die Dynamisch

Woher bekomme ich meine Daten

Wie nutze ich meine Business Logik

API/Backend

Wo ist/sind meine Datenbank(en)

Welche Datenmanipulationen sind notwendig

Wie sind meine Schnittstellen definiert
-> Datenmodelle

Wetter APP

Der user will das Wetter an Adresse X finden	Wir brauchen eine API die die Möglichkeit hat, eine Adresse als Eingabe zu bekommen und das Wetterobjekt mit folgenden Informationen als Response liefert:

Adresse

Wetter Anzeigen

Wettervorhersage

Temperatur

Datum

Wind

Startseite

Der Nutzer trägt seine Adresse ein und klickt den Button

Wetterseite

Der Nutzer sieht das Wetter

Klausurprojekt

Projekt Booking Software

Anforderungen

- Eine Buchungssoftware für einen beliebigen Use Case (Restaurant, Sportkurs, Vorlesungsraum, Meetingraum, etc.), die mit NEXTJS umgesetzt wurde
- Die Buchungssoftware besteht aus mindestens 3 Seiten
  - Eingabe der Nutzerinformationen
    - Eingabefelder für z.B. Name, Telefon und Email
  - Kalenderübersicht für die Buchungsdurchführung
    - Kalender/mögliche Buchungen, Buchung bestätigen, Abbrechen
  - Bestätigungsseite für die Buchung
    - Zurück zur Startseite, Übersicht über die Buchung, Buchung bearbeiten
- Die Buchungssoftware nutzt eine RestAPI um die Daten vom Frontend in ein Datenmanagement im Backend zu übertragen
  - Die Daten werden lokal in der API oder in einer Datenbank gespeichert
  - Die API verfügt über Funktionen zum Lesen, Schreiben und Bearbeiten und Löschen von Buchungen

Bewertungskriterien

Klausurprojekt

40 %

Paper Prototype	20
Klassendiagramm	20
Datenstruktur	20
Flow Diagram	20
Konzept	20

60 %

Struktur	10
Min 1 mal FlexBox und min 1 mal Grid in min 1 externen CSS-File und min 1 mal einer Style Variable	10
Conditional & List Rendering	10
Serverside Action	10
use State	10
Min. 1 mal Serverside und 1 mal Clientside Rendering	10
Props & Two Way Binding	10
API Implementierung mit get, post, put & delete	10
Codequalität	10
Funktion	10

Formale Richtlinien

Klausurprojekt

- Abgabe des Projektes über Github **und** Moodle (Dokumentation (PDF) und Code)
- Der Projektname ist eure Matrikelnummer
- Die Dokumentation wird **IMMER ALS PDF** abgegeben/Name des PDFs ist die Matrikelnummer (GitHub **und** moodle)
- Projekt in Github: Commithistorie muss vorhanden sein (mindestens 10 commits, die den Fortschritt zeigen)
- Jeder gibt sein eigenes Projekt (Dokumentation und Code) ab, welches er zu 100% selbst erstellt
- ABGABE: 16.08.2024 23:59 Uhr
- Github: **private Repository** und wdskiWebDev@jonas-heuer.com als Contributor hinzufügen (Nutzername: WDSKI)

Usefull Links

- https://www.redhat.com/en/topics/api/what-is-a-rest-api
- https://restfulapi.net/http-methods/
- https://aws.amazon.com/de/compare/the-difference-between-mysql-vs-postgresql/
- https://css-tricks.com/snippets/css/complete-guide-grid/
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- https://nextjs.org/docs
- https://www.ibm.com/de-de/topics/rest-apis
- https://react.dev/
- https://www.freecodecamp.org/news/a-thorough-introduction-to-distributed-systems-3b91562c9b3c/
- https://www.spiceworks.com/tech/cloud/articles/what-is-distributed-computing/#:~:text=A distributed system is broadly,architecture and peer-to-peer
- https://scalac.io/synchronous-vs-asynchronous/
- https://www.distributed-systems.net/index.php/books/ds4/
- https://fusionauth.io/articles/oauth/modern-guide-to-oauth