4. Full-Stack Entwicklung

Wie kommen Daten aus einer DB in eine App?

Applikationssoftware wird oft in einen **Frontend-Teil** und einen **Backend-Teil** unterteilt. Während das Frontend sich hauptsächlich um das **GUI** (Graphical User Interface) und somit den User direkt kümmert, verrichtet das Backend wichtige Arbeiten im Stillen. So ist das Backend meist für die sogennante **Persistenz** zuständig, d.h. es kümmert sich darum, wie und wo Daten gespeichert werden, wenn die App gerade nicht läuft. Das Backend ist mit dem Trend zur "Datensammelwut" und zur Auslagerung der Daten in die Cloud in den letzten Jahren immer wichtiger geworden.

Frontend

Das Frontend ist nah beim Benutzer. Bei einer Client-Server-Lösung ist es der **Client-Teil**, also der Teil der beim Benutzer in einem Browser oder als eigenständige App läuft. Wenn es auf persistente (Daten und Zustände über lange Zeit speichernd) Daten zugreifen will, greift es meist über eine Schnittstelle, welche auch **API** (Application Interface) genannt wird, auf ein Backend zu.

Beim Frontend kommen die Sprachen HTML, CSS, JavaScript, React, Angular, Vue.js, ASP.NET und eine Vielzahl von Frameworks zum Einsatz.

Backend

Das Backend speichert und verwaltet, meist mithilfe einer Datenbank, die von den Usern bereitgestellten Daten. Das Backend läuft meistens auf einem Server der Firma oder in der Cloud. Bei einer Client-Server-Lösung ist das Backend die Software, die auf dem Server läuft.

Beim Backend kommen die Programmiersprachen PHP, ASP, Javascript und Systemdienste wie Web-Server (Apache, Nginx oder node.js) und Datenbanken wie MySQL, MariaDB, MongoDB und FireBase zum Einsatz.

Full-Stack oder Solution Stack

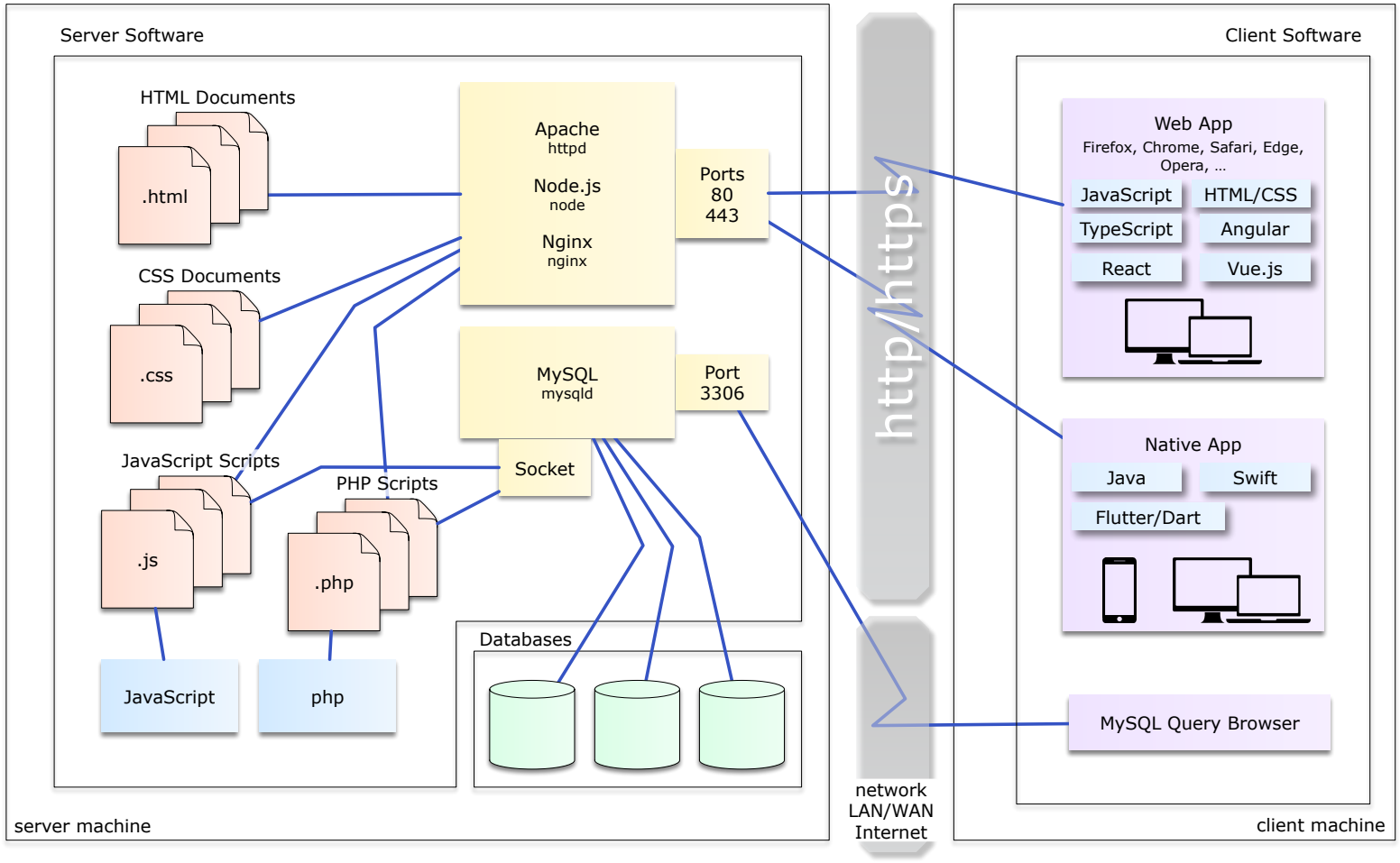
Nimmt man das Frontend und das Backend zusammen, so hat man einen Full-Stack oder Solution Stack.

Er beinhalted also die komplette Applikation vom GUI auf dem Handy bis zur DB in der Cloud.

Programmierer:innen, die den ganzen Pfad, also alle Sprachen und Technologien vom Frontend bis zum Backend beherrschen, nennt man Full-Stack-Entwickler:innen. Das sind im Moment die bestbezahlten Jobs der IT.

Gängige Solution Stacks sind:

- LAMPP: Linux, Apache, MariaDB, PHP, Perl
- MEAN: MongoDB, Express, Angular, Node
- MERN: MariaDB, Express, React, Node
- WISA: Windows Server, Internet Information Services, Microsoft SQL Server, ASP.NET



Projektaufgabe jukeStack

In diesem Projekt geht es darum, eine Web-App zu erstellen, mit welcher man eine elektronische JukeBox mit NFT-Songs (Non-Fungible Tokens) benutzen und verwalten kann. Ihr Vorgänger hat bereits eine Analyse und Teile des Designs (siehe jukeStack.pdf auf dem Portal) erstellt.

Aufgaben:

1. Tun Sie sich zu zweit zusammen.
2. Arbeiten Sie das Tutorial "Beginners CRUD Tutorial" https://youtu.be/re3OIOr9dJI 1:1 durch.
3. Richten Sie auf Trello ein neues Projekt **jukeStack_*Name1Name2*** ein, auf welches beide Teampartner:innen plus Ihre Lehrperson (Einladung versenden) Zugriff haben. Planen Sie alle Arbeiten fortan auf Trello, indem Sie für jede Aufgabe unter **ToDo** ein Kärtchen erstellen. Sobald jemand die Aufgabe in Angriff nimmt, trägt er im Kärtchen seinen **Namen** ein und verschiebt das Kärtchen nach **InProcess**. Jeder Task der fertig gestellt ist, wird dann nach **Done** verschoben.
4. Richten Sie auf GitHub ein neues Projekt **jukeStack_*Name1Name2*** ein auf welches beide Teampartner:innen plus Ihre Lehrperson (Einladung versenden) Zugriff haben. Pushen Sie Ihr Projekt fortan inkl. Doku regelmässig auf GitHub.
5. Erstellen Sie eine Dokumentation **jukeStackDoc_*Name1Name2*.doc*.** Übernehmen Sie alles was Ihr Vorgänger schon gemacht hat (siehe **jukeStack.pdf** auf dem Portal) in dieses Dokument und pushen Sie auch dieses nach jeder Änderung auf GitHub.
6. Erstellen Sie für jukeStack ein Datenbankschema (die DB soll **jukeStackDB_*Name1Name2*** heissen) und daraus ein SQL-Script, welches die DB erzeugt. Das Schema fügen Sie in **jukeStackDoc** ein. Erstellen Sie mit dem Script die DB auf i-kf.ch.
7. Erstellen Sie in **jukeStackDoc** auch eine Liste mit Testfällen, die Sie später für die Tests benötigen. Achten Sie darauf, dass Sie je mindestens einen Testfall für das Hinzufügen, einen für das Ändern und einen für das Löschen (CRUD) einer Entität jedes modifizierbaren Entitätstyps haben.

7. Implementieren Sie, vorläufig noch ohne Login, die Web-App mit einem MERN (MySQL, Express, React, Node) Stack. Legen Sie den Fokus zunächst auf das Backend. Das Frontend muss überhaupt nicht schön sein. Es muss nur funktionieren. Denken Sie daran, dass Sie via GitHub und Trello im Team arbeiten - also immer schön Pushen und Kärtchen für Kärtchen abarbeiten)!
8. Erstellen Sie von Ihrem Projekt ein Screen-Video **jukeStackVid_*Name1Name2*.mp4**, in welchem Sie anhand der Testfälle aufzeigen, dass Ihre Web-App korrekt funktioniert. Laden Sie das Video spätestens bis zum **Abgabetermin I** aufs Portal.
9. Erstellen Sie Layout-Mockups für das Login, die Registrierung und die Hauptseite und fügen Sie sie in die Doku ein.
10. Erstellen Sie nun das Login und die Registrierung.
11. Verschönern Sie das GUI (Frontend) und machen Sie von jedem Layout einen Screenshot welchen Sie in die Doku im Kapitel Benutzeranleitung einfügen.
12. Geben Sie die Doku bis zum **Abgabetermin II** auf dem Portal ab.

Abgabetermine:

Abgabetermin 1: Do **12.01.2023 24:00** Uhr

Abgabetermin 2: Do **26.01.2023 24:00** Uhr

Noten:

Die Aufgabe wird nach dem Termin 1 mit einer ganz zählenden Note bewertet.

Bewertet werden die Kriterien: Trellobenutzung, Datenbankschema, DB-Script, Backend, Funktionalität