

PROJEKT 1

Projekt 1: Notizen Verwalten

■ Ziel von Projekt 1

- Grundlagen festigen von CSS/JS/HTML

■ Aufgabe

- Erstellen einer Notizenapplikation

The screenshot shows the 'Note Pro' application interface. At the top, there's a browser-like address bar with 'http://www.notes.ch'. Below it, a 'Create new Note' button and a 'BlackWhite-Style' dropdown menu are visible. A navigation bar contains four buttons: 'By finish Date', 'By created Date', 'By Importance' (which is highlighted), and 'Show finished'. The main content area displays a list of notes categorized by date. Under 'Nächsten Mittwoch', there's a note 'CAS FEE Selbststudium / Projekt Aufgabe erledigen' with two lightning bolt icons. Under 'Heute', there's a note 'Einkaufen' with one lightning bolt icon, and a note 'Mami anrufen' with no icons. Each note has a text input field, a dropdown arrow, and an 'Edit' button. The 'Finished' checkbox is checked for the first two notes and unchecked for the third. The 'Show finished' button is also present.

The screenshot shows the 'Note Pro' application interface in 'Edit Note' mode. At the top, there's a browser-like address bar with 'http://www.notes.ch'. Below it, a 'Title' input field and a 'Beschreibung' (Description) text area are visible. The 'Beschreibung' area contains the text 'HTML für die note App erstellen. CSS erstellen für die note App. [...]'. Below the description, there's a 'Wichtigkeit' (Importance) section with five lightning bolt icons, where the first two are filled and the next three are empty. Below that, there's an 'Erledigt bis:' (Due by) section with a date input field showing '//' and a calendar icon. At the bottom, there's a 'Speichern' (Save) button and a 'Cancel' button.

Projekt 1

- **Startet heute.**
Aktion heute: Eintrag in Einschreibeliste (Name und GitHub URL)
<https://docs.google.com/spreadsheets/d/1hB0OCNTtcwiYkbg6TZUzn2dLUZEFKCqtjVZqGEugGk/edit?usp=sharing>
- **Bewertung: Pass/Fail (must pass)**
- **Arbeit alleine, aber öffentlich (Github Account in Liste eintragen)**
- **Abgabetermin 27.06.2018 – Mitternacht**
 - Branch erstellen mit dem Namen «Abgabe»
 - E-Mail erstellen mit folgendem Inhalt:
 - Subject: [CAS FEE] Abgabe {{Gruppennummer}}
 - Link zum Branch
 - Ein ReadMe auf GitHub, falls dieses notwendig ist.
- **Bei anderen Projekten nachzuschauen ist erlaubt, die Lösung muss aber eine Eigenleistung sein**
- **Erfolgreiche Abgabe von Projekt 1 ist Voraussetzung zum Start von Projekt 2**
 - Tipp: Bei «Problemen» früh melden und nicht bis zum Schluss warten

Projekt 1 – Einschränkungen

- Das Projekt sollte gut strukturiert sein aber weitgehend auf die Nutzung von Frameworks verzichten
 - Nutzung eines SPA Frameworks (Angular oder React+...) **nicht** erlaubt
 - Nutzung des MVC Patterns trotzdem sinnvoll
 - Nutzung von Layout / Styling-Framework wie Bootstrap nicht erlaubt
 - Flexbox und Grid nutzen
 - Nutzung von simpler Templating Engine (z.B. Handlebars) ist verlangt.
 - Nutzung von jQuery ist erlaubt aber nicht verlangt
- Ältere Browser müssen nicht unterstützt werden.

Projekt 1 – «Bewertung»

- **Das Projekt 1 wird in 5 Kategorien angeschaut. Jede Kategorie muss erfüllt sein.**
 - Funktionsumfang
 - Architektur
 - JS Qualität
 - CSS Qualität
 - HTML Qualität
- **Es wird ein generelles Feedback über alle Projekte geben.**
- **Bei größeren Problem in einem Projekt erhält man 2 Wochen Zeit um diese zu beheben.**

Projekt 1 – «Bewertung»

■ Funktionsumfang

- Der Funktionsumfang ist in den Wire-Frames dargestellt. Diese beinhalten u.a.
 - Anzeigen, editieren und erfassen von Notizen
 - Sortieren von Notizen
 - Filtern von „abgeschlossenen“ Notizen
 - Abspeichern der Daten auf dem Server
 - Wechseln des Styles

■ Allgemeine Code Guidelines

- Unterstützung modernen Browsers & Features (Ältere Browser können vernachlässigt werden)
 - CSS3+, HTML5+, ES6+
- Saubere Trennung von Struktur (HTML/HBS), Logik (JS) und Darstellung (CSS)
- Sauberen Code
 - DRY: Kein Copy-Paste-Code (auch keine 'ähnlicher Code'!)
 - Keine langen Methoden
 - Sprechende, konsistente Benennung von Variablen
- Kein CSS / JS im HTML
- Übersichtliche Projekt-Struktur

■ Architektur

- REST: Server und Client kommunizieren über JSON
- Client:
 - Server-Calls nur im Service Layer
 - Routing/Event-Handling nur im Controller
 - Rendering/DOM Manipulation nur in der View
 - View und Controller können im gleichem File definiert werden.
 - Kein HTML Zusammenbasteln aus String => Handlebars verwenden
- Server
 - Memory-Storage- / DB-Zugriffe nur im Service
 - Controller stellt Actions/Request-Handlers zur Verfügung
 - Router: Verknüpfung von Routen und Actions/Request-handlers

■ JavaScript

- Nutzung einer Template Engine auf dem Client (z.B. Handlebars)
- Keine Console Pollution
- Kein auskommentierter Code
- Kein «global namespace pollution»

■ CSS

- FlexBox
 - keine Float-, Table- oder Inline Layouts
 - keine unnötigen Klassen & ID's (Elemente über Struktur/Name selektieren)
- Inline Styles & Inline-Style-Klassen sind nicht erlaubt

■ HTML

- Korrekter Einsatz von Semantischem HTML
- Sinnvolle HTML Validation

Projekt 1 – Ablauf

Woche	Aufgabe-Grob	Details
1	Projekt auf Github anlegen und in Liste eintragen. Codewars-Account erstellen. HTML Gerüst erstellen für die WireFrames und Beginn CSS.	Ändern der Wireframes ist erlaubt.
2	«Create New Note» auf der Detail-Seite implementieren. Die Daten in den «LocalStorage» einpflegen. Navigation zwischen den beiden HTML-Seiten. Auf der Hauptseite die Anzahl darstellen.	
3	HTML & CSS mit Grid / Flex-layout Style Switcher implementieren.	Einen zweiten sinnvollen Style ausdenken.
4	Hauptseite ausprogrammieren: Anzeigen der Einträge / Filtern / Sortieren Handlebars verwenden für das Rendern der Einträge. Detailseite ausprogrammieren: Erfassen / Editieren	Flex-layout im Projekt einsetzen.
5	JavaScript optimieren. Patterns anwenden. Nutzen von Klassen für die Datenhaltung.	(Revealing) Module Pattern für die "Datenklassen" erstellen. IIFE anwenden.
6	Client Modularisierung fortführen. Node-Module erstellen zum Verwalten der Daten auf dem Server.	Bonus: Neue Einträge sollen auf andern Browser sichtbar werden. z.B. durch Polling.
7	Die REST API vom Server implementieren. Diese im Client anbinden.	
8+9	Finalisieren & Abgabe	