Spengergram

The last social media app you'll (n)ever need 😉

`From 0 to Hero: Master Software Engineering on Your Learning Journey with Spengergram!`

Beschreibung

Spengergram ist eine kompakte Social-Media-App, die es Nutzern ermöglicht, sich zu registrieren, Fotos zu teilen und diese auf einer Timeline zu liken. Zusätzlich bietet die App Funktionen zum Verwalten von Freundschaften und ein separates Admin-Dashboard. Als vereinfachte Version von Instagram dient Spengergram als praxisorientiertes Lernprojekt, in dem Studierende sowohl grundlegende als auch fortgeschrittene Techniken der App-Entwicklung im realen Einsatz erlernen.

Motivation

Das Ziel des Projekts ist es, den Studierenden eine praxisnahe Erfahrung zu bieten, indem sie den gesamten Entwicklungsprozess einer App von der Konzeption bis zur Produktionsreife durchlaufen. Dies ermöglicht es ihnen, sowohl theoretisches Wissen als auch praktische Fähigkeiten in der Entwicklung von Web- und Mobile-Anwendungen anzuwenden. Das Projekt dient zudem als Musterbeispiel, das die Studierenden für Diplomprojekte und berufliche Vorhaben nutzen können.

Vision

Die grundlegende Vision von Spengergram ist es, POS, WMC, PRE und BWM zu integrieren, um Studierenden anhand eines realen Softwareprojekts eine umfassende, praxisnahe Ausbildung in Softwareentwicklung und interdisziplinärer Zusammenarbeit zu ermöglichen, sie bei ihren Diplomprojekten zu unterstützen und bestmöglich auf das Arbeitsleben in Softwareprojekten vorzubereiten.

Lehrmethodik

- **Git-Book:** Die Lehrinhalte werden in einem Git-Book bereitgestellt, das den Studierenden einen strukturierten Überblick über die Lehrziele, Aufgaben und Ressourcen bietet.
- **Git-Classroom:** Die Studierenden arbeiten an den Lehrzielen und reichen ihre Lösungen über Git-Classroom ein.
- **Folien:** Die Lehrinhalte werden in zusammengefasster Form in Folien präsentiert.
- **Etwa 10 große Lehrziele pro Semester:** Die Studierenden arbeiten an umfangreichen Lehrzielen, die breit gefächerte Kompetenzen abdecken, incl. Bonus-Lehrziele.
- **Flipped Classroom:** Der Foliensatz für die kommende Woche wird jeweils in der Vorwoche ausgegeben, um die Vorbereitung zu erleichtern.
- **Individuelle Vorbereitung:** Die Vorbereitung der Studierenden wird als Teil der fortlaufenden Bewertung ihrer Mitarbeit gewertet und mit Quizzen überprüft.
- **Interaktive Entwicklung:** Jede Woche beginnt mit der Vorstellung des Lehrziels, das zunehmend gemeinsam entwickelt wird, wobei der Anteil der Lehrenden schrittweise abnimmt.

- **Abschluss durch Studierende:** Der Abschluss jedes Lehrziels erfolgt durch die Studierenden; die Beurteilung geschieht über Pull-Requests, um praktische Softwareentwicklungskompetenzen zu bewerten.
- **Nachträgliche Auflösung:** Die Auflösung und Besprechung der Lehrziele erfolgt nachträglich, um ein tiefgreifendes Verständnis und die Reflexion über die gelösten Aufgaben zu gewährleisten.

Bewertung

- **Wöchentliche Quizze:** Die Quizze dienen der Überprüfung des Wissensstands und der Vorbereitung der Studierenden. Sie fließen in die Bewertung der Mitarbeit ein.
- **Wöchentliche Pull-Requests:** Die Studierenden müssen jede Woche einen Pull-Request einreichen. Die Qualität dieser Pull-Requests ist ein zentrales Bewertungskriterium und reflektiert die kontinuierliche Mitarbeit und technische Umsetzung des Projekts.
- **Praktische Leistungsüberprüfung (PLÜ):** Zusätzlich zur laufenden Bewertung der Pull-Requests findet pro Semester eine PLÜ statt, die darauf abzielt, das erlernte Wissen und die praktischen Fähigkeiten der Studierenden zu bewerten.
- **Referate und aktive Teilnahme:** Als ergänzende Bewertungsoption können die Studierenden Referate zu spezifischen Themenbereichen halten. Zudem wird die aktive Teilnahme am Unterricht bewertet, um Engagement und Beteiligung im Lernprozess zu fördern.
- **Open Source:** Spengergram wird als fertige Implementierung den Studierenden zur Verfügung stehen. Auf freiwilliger Basis können sie Pull Requests mit eigenen Erweiterungen einreichen. Die besten 2 (oder 3) Pull Requests werden Ende des Jahres accepted und fließen in den Unterricht des Folgejahres ein. (Quasi ein Contest und damit eine enorme zusätzliche Motivation für die Studierenden). Der POS-Unterricht wird Open Source wie es in der Praxis gelebt wird.

Lehrziele POS

Wintersemester (Matura Main-Facts, VWA Main-Facts)

- *Umsetzung einer Todo-App*
- C# Basics / .NET Basics und Unterschiede zum bekannten Java
    - **Wertetypen / Referenztypen [GRUNDLEGEND]**
    - **Collections (Übersicht, foreach-Keyword) [GRUNDLEGEND]**
    - **Anonymous Types [GRUNDLEGEND]**
    - **Delegates [GRUNDLEGEND]**
    - **Lambdas [GRUNDLEGEND]**
    - Partial Types
    - Überladene Operatoren
    - Initializer
    - Vererbung
    - Spread-Operator ([..])
    - Nullable Feature
    - Return Tuples
    - Extension Methods
- DDD
    - **Aggregates, Entities, Value Objects [GRUNDLEGEND]**
    - Validations

    - Logik
    - DTO-Mapping
    - Rich Typed IDs
- **LinQ: Übungen und praktische Beispiele [GRUNDLEGEND]**
- OR-Mapper:
    - **Code First [GRUNDLEGEND]**
    - **Data Annotations [GRUNDLEGEND]**
    - Database First
    - Fluent API
- 3-Layer Architecture
- Basic-Patterns:
    - **Repository [GRUNDLEGEND]**
    - Builder
- **Unit Testing: Simpel [GRUNDLEGEND]**
- **Data Validation: Simpel [GRUNDLEGEND]**
- **API-Design: URI, Controller, Http-Methods, Status Codes [GRUNDLEGEND]**

Sommersemester (Die ersten Teile werden bereits im WS gestartet)

- *Umsetzung der Spengergram-App*
- Clean Architecture
- Authentication: Basics, OAuth, JWT, Session-Token, Auth-Provider
- Authorization: Role Based und Policy Based Authentication. Umsetzung mit Identity Framework
- Fluent API: Fluent API udn Chaining selbst erstellen können.
- CQS / CQRS
    - **CQS [GRUNDLEGEND]**
    - CQRS: CQS in praktischer Umsetzung, CQRS wenn Zeit bleibt.
- Event Sourcing: Mit RabbitMQ als Message Broker (als Docker-Container)
- **S.O.L.I.D.: Wird theoretisch besprochen und vorgezeigt. Hier sind eigene Aufgaben zu lösen. [GRUNDLEGEND]**
- **Inversion of Control: Dependency Inversion, Dependency Injection [GRUNDLEGEND]**
- Advanced-Patterns (Mediator, Fluent API, Template, ...)
- Composability: Wird anhand CQS gezeigt
- **Separation of Concern [GRUNDLEGEND]**
- **Action Filter: Erstellung und Anwendung [GRUNDLEGEND]**
- Middleware: Erstellung und Anwendung, Unterschiede zu Action Filtern
- Caching
- Logging / Tracing: Umsetzung direkt und mit APO
- **Data Validation: Validierung in allen Schichten inkl. Datenbank [GRUNDLEGEND]**
- TDD:
    - **Test Driven [GRUNDLEGEND]**
    - BDD
    - Mocking
    - Fluent Assertion
- Integration Testing: Controller Testing mit Mock-MVC
- OpenAPI Specification (Konsistente BE/FE Validierung)
- HATEOAS

Lehrziele WMC

Wintersemester (Matura Main-Facts)

- *Umsetzung einer Todo-App (Cross platform: Web/Mobile)*
- Introduction Web/Native App Development
  - **Native vs Hybrid vs Progressive Web Apps [GRUNDLEGEND]**
- JavaScript Recap
  - **Promises, Async/Await [GRUNDLEGEND]**
  - **Spread Operator, Rest Parameter, Destructuring [GRUNDLEGEND]**
  - **Array-Methods (map, filter, reduce, ...) [GRUNDLEGEND]**
  - **Classes vs Prototypes [GRUNDLEGEND]**
  - **Functions, Lambdas [GRUNDLEGEND]**
  - **Fetch-API [GRUNDLEGEND]**
  - Object-Methods (Object.keys, Object.values, ...)
  - ES6 Modules
  - CommonJS vs UMD vs ES6 Modules
  - JSON
- Typescript Foundation
  - **Types, Interfaces, Enums, Generics [GRUNDLEGEND]**
  - Intersection, Union types
  - Utility Types, Type Guards
  - .d.ts Files (Declaration Files)
  - .tsconfig File (Compiler Options)
- Sass Foundation
  - **Variables, Mixins [GRUNDLEGEND]**
  - BEM-Style
- Setup Workspace for Web/Mobile Development
  - **Setup Web Development Environment (node, npm, ..) [GRUNDLEGEND]**
  - **Setup Android Development Environment (Gradle, JDK, Android Studio) [GRUNDLEGEND]**
  - Setup iOS Development Environment (Xcode)
- React Components Foundation
  - **JSX Lifecycle, State, Props, Events, Hooks [GRUNDLEGEND]**
  - **Component-Libraries (Native Base) [GRUNDLEGEND]**
  - **Stateful vs Stateless Components [GRUNDLEGEND]**
  - Component-Patterns (Higher Order, Render-Props)
- React Routing Foundation
  - **Frontend vs Backend Routing [GRUNDLEGEND]**
  - **URL-Aufbau [GRUNDLEGEND]**
  - **Setup Basic Routing [GRUNDLEGEND]**
  - Route Params, Query Params
  - Component Routing
  - Programmatic Routing
- HTTP Foundation
  - **Fetch API vs Axios [GRUNDLEGEND]**
  - **HTTP Methods, Headers, Body, Status Codes [GRUNDLEGEND]**
  - Mock APIs (JSON-Server)

    - Error Handling
- Forms Foundation
    - **Basic Form Setup [GRUNDLEGEND]**
    - **Form Validation (Client-Side, Server-Side) [GRUNDLEGEND]**
    - Controlled vs Uncontrolled Components
    - Form Libraries (Formik)
- In-App Notifications Foundation
    - **Snackbars [GRUNDLEGEND]**
    - **Modals [GRUNDLEGEND]**
- Testing Foundation
    - **Basic Unit Testing [GRUNDLEGEND]**
- Style Guides
    - Airbnb React/JSX Style Guide

Sommersemester (Die ersten Teile werden bereits im WS gestartet)

- *Umsetzung der Spengergram-App (Cross platform: Web/Mobile)*
- Design System
    - **Figma [GRUNDLEGEND]**
    - **UI/UX Regeln, Human Interface Guidelines (VWA) [GRUNDLEGEND]**
    - Storybook
    - Design Tokens
    - Theming
- NX Mono Repository
    - Multi-Apps
- NX Lib Architecture
    - **Shared, Feature, UI, Data, Util [GRUNDLEGEND]**
    - Sharing Code between Web and Mobile
- Shell Patterns
    - App Shell Pattern
    - **Feature Shell Pattern [GRUNDLEGEND]**
- Advanced React Components
    - **Container vs Provider [GRUNDLEGEND]**
    - **List Components [GRUNDLEGEND]**
    - Grid Components
    - Image Gallery Components
    - Bottom Sheet Components
    - Infinity Scroll
    - Composition vs Inheritance
- Advanced React Routing
    - **Unterschiede im zwischen Web und Mobile [GRUNDLEGEND]**
    - Verwendung von Navigation Stacks in Mobile-Apps
    - Tabs und Drawer Navigation in Mobile-Apps
    - Route Animationen
    - Dynamic Routing
    - Lazy Loading / Preloading
    - Deep Linking

- Advanced HTTP
  - **Authentication (Basic, Bearer) [GRUNDLEGEND]**
  - Cookies vs Tokens
  - File-Upload, File-Download
  - Interceptors
  - Caching Strategies
  - Retry Strategies
- Statement Management (VWA)
  - **Component State vs Application State [GRUNDLEGEND]**
  - **Redux-Pattern Basic (Actions, Reducers, Store) [GRUNDLEGEND]**
  - Redux-Pattern Advanced (Thunks, Saga)
- Advanced Navigation
  - **Top, Bottom, Side Navigation [GRUNDLEGEND]**
  - Swipe Navigation
- Native Features
  - Camera, Gallery
  - Push-Notifications
  - Storage, Battery, Geolocation
  - Device Orientation
- Advanced Testing
  - E2E Tests
  - Jest, Playwright, Maestro
- User Feedback
  - **Progress Indicator [GRUNDLEGEND]**
  - Skeletons
  - Blurred Hashes
  - Thumbnails
- Application Observability
  - **Debugging [GRUNDLEGEND]**
  - Logging
  - Monitoring
  - Error Tracking (Sentry)
- Offline First
  - **Was bedeutet Offline First? [GRUNDLEGEND]**
  - Cache-Strategies
  - Sync-Strategies
  - Background Sync
- Performance Optimization
  - Profiling
  - Performance Tuning
- Internationalization (i18n)
- Web Security / Mobile Security (VWA)
  - **SOP, CORS, CSRF, XSS [GRUNDLEGEND]**
  - Secure Storage
  - Secure Communication
  - OWASP Top 10

- Mobile App Stores (BONUS Lehrziel)
  - App Submission Process
- Writing Native Plugins (BONUS Lehrziel)
  - Schreiben von nativen Plugins für Web/Mobile

Spengergram-App

- Die Spengergram-App beginnt nach der Todo-App und baut auf den erworbenen Kenntnissen und Fähigkeiten auf.
- Die Umsetzung beginnt und erfolgt mittels bekannter moderner Softwareentwicklung- und DevOps-Praktiken.

DevOps-Komponenten

- **Infrastruktur-Automatisierung**: Einsatz von Infrastructure as Code (IaC).
- **Versionskontrolle und Code-Management**: Einrichtung von Git-Repositories, Branching-Strategien und Code-Review-Prozessen.
- **Continuous Integration und Continuous Deployment (CI/CD)**: Konfiguration automatisierter Build- und Deployment-Pipelines mit Tools wie GitHub Actions.
- **Containerisierung und Orchestrierung**: Verwendung von Docker und Kubernetes zur Verwaltung der Anwendungscontainer.
- **SecDevOps**: Secrets Management, Vulnerability Scanner
- **Load Testing:** Durchführung von Lasttests, um die Leistungsgrenzen der Anwendung zu ermitteln und Engpässe zu identifizieren.
- **Monitoring:** Kontinuierliche Überwachung der Anwendungsleistung und Systemressourcen, um Probleme frühzeitig zu erkennen und die Systemstabilität sicherzustellen.
- **Logging:** Protokollierung von System- und Anwendungsereignissen, die für die Fehlerdiagnose und die Performance-Analyse entscheidend sind.

DevOps-Voraussetzungen

- **Moderne Softwareentwicklungs-Praktiken:** Nutzung von Tools wie `Jira` und `Confluence` für die Organisation und Kollaboration im Team. Einsatz von `Miro` und `Figma` zur Unterstützung der visuellen Zusammenarbeit und Designprozesse. Anwendung von `User Stories`, ergänzt durch Entwicklungspraktiken wie `Test-Driven Development (TDD)` und `Behavior-Driven Development (BDD)`.

Deployment

- **Cloud:** Es wird ein Deployment auf einem Cloud-Provider (z.B. AWS, Azure, Google Cloud) angestrebt.
- **Eigener Server**: Oder ein Deployment auf einem eigenen Server (z.B. Hetzner, ...)
- **Deployment-Strategien**: Feature-Flags, Canary-Deployment, A/B-Testing, ...

Features

- Registrierung
- Anmeldung / Abmeldung
- Anzeigen der Timeline mit Posts als Infinite Scroll
- Filtern, Sortieren und Suchen von Posts

- Post erstellen, liken
- Freundschaftsanfragen senden, annehmen, ablehnen
- Kleines Admin-Dashboard für Statistiken und Benutzerverwaltung
- BONUS: Push-Notifications
- BONUS: Post Kommentare
- BONUS: Chat-Funktion

Lehrziele 1 bis 10 pro Semester / Fach

Werden über den Sommer ausgearbeitet.

Lehrziel Struktur

1. Titel
2. Kontext und Bedeutung des Themas
3. Problemstellung / Lösung
4. Theoretischer Hintergrund
5. Praktische Beispiele
6. Aufgabenstellung für das Pull Request
7. Quiz
8. Reflexion

Appendix: Matura Themen

1. Architektur von Web- und Mobile-Anwendungen

- Frontend/Backend Aufteilung
- REST, RESTful API
- Komponentenkonzept (Clientseitig)
- Controller, Services im Backend
- Testen

2. Datenmanagement und Persistenz in Web- und Mobile-Anwendungen

- CRUD Operationen und deren Abbildung im Frontend
- CRUD Operationen und deren Abbildung im Backend
- Validierung (clientseitig, serverseitig)

3. Development, Deployment und Betrieb von Web- und Mobile-Anwendungen

- Containertechnik
- Cloudprovider
- Continuous deployment

4. Frameworks im Bereich der Web- und Mobile-Anwendungen

- Komponentenkonzept (Clientseitig) in Hinblick auf Vue.js, Angular, ...
- Entwicklungswerkzeuge
- State Management

- Andere Themen aus Vue, Angular, … die behandelt wurden.

5. Sicherheit in Web- und Mobile-Anwendungen

- Authentication und Authorization
- Implementierung im Backend
- Implementierung im Frontend
- Angriffsvektoren im Bereich der Webapplikationen

6. User Experience & User Interfaces in Web- und Mobile-Anwendungen

- Responsive und Adaptive Design
- Auszeichnungssprachen für die Gestaltung
- Formulargestaltung
- Barrierefreies Webdesign