

Abschlusspräsentation

Projekt







Projektmanagement

Scrum Lite

-  Organisation über Gitlab
-  Arbeitspakete in Sprints (Scrum Lite)

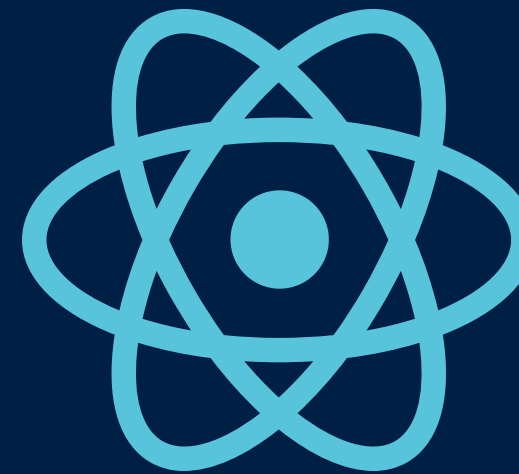
Code Quality

-  Gitflow mit »Protected Branching«
-  CI-/CD-Pipeline
-  Merge-Requests mit Reviews
-  »Pair programming«

Frontend Technologien



Tailwind CSS



React



Vite



TypeScript



TanStack

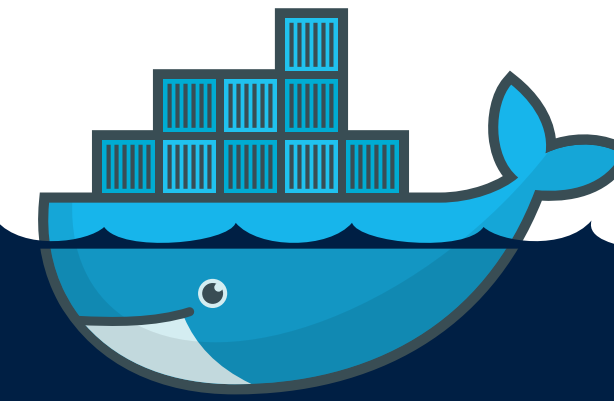


React Query

Backend Technologien



Deployment

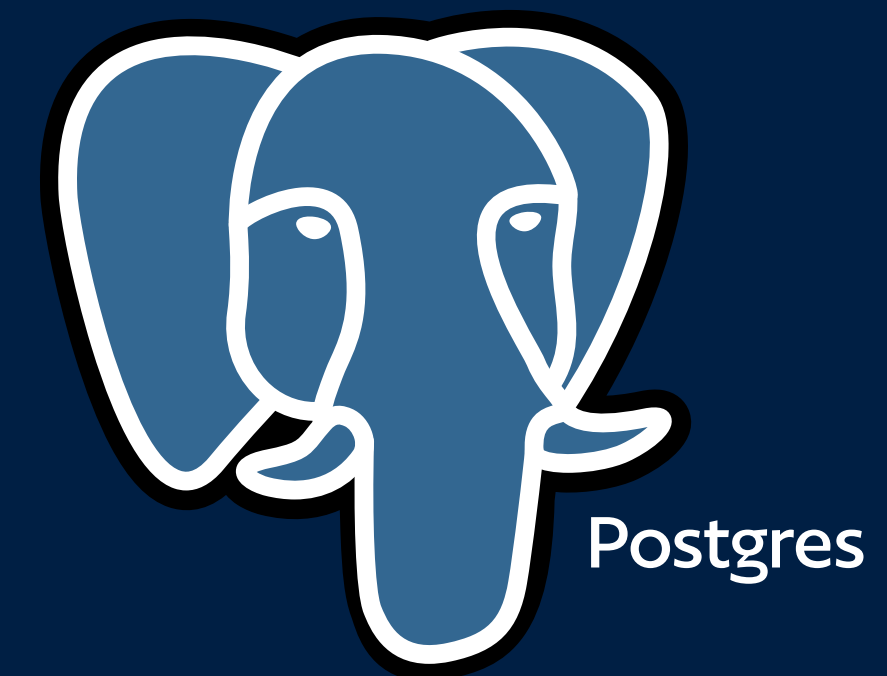


docker

Deployment: docker-compose

Images

-  Application
-  Postgres
-  Mail (Mailpit)



Staging

-  Application (<https://staging.hs-flensburg.dev>)
-  Mail (<https://mail.hs-flensburg.dev>)

Deployment mit CI und docker-compose



Funktionalitäten

Erstellung von Klassenräumen

- 🐙 Dozierende können Klassenzimmer erstellen und Studierende per E-Mail mit personalisiertem Link einladen.
- 🐙 Nach Beitritt wird automatisch eine GitLab-Gruppe erstellt.

Funktionalitäten

Erstellung von Teams

-  Studierende können Teams beitreten oder eigene erstellen; Einzelarbeit ist auch möglich.
-  GitLab-Untergruppen werden für Teams mit entsprechenden Berechtigungen angelegt.

Funktionalitäten

Erstellung von Aufgaben

- 🐙 Dozierende können beliebig viele Aufgaben erstellen und optional Abgabetermine festlegen.

Funktionalitäten

Optional: **Bewertungssystem**

- 🐙 Automatisches Bewertungssystem überprüft Code per Tests, manuelles System ermöglicht individuelle Bewertungen und Feedback.

Sprints 1-4

Kickoff

25.10.23 - 8.11.23

- Ziel des Projekts im Kickoff-Meeting mit Dozenten erörtert
- Technologien und organisatorische Aspekte (Termine, Rollenverteilung) festgelegt
- Teammitglieder sollen sich vorab mit Technologien vertraut machen
- Erster UI/UX-Prototyp mit Figma erstellt

Sprint 1-3

Die Grundlagen

- GitLab API testweise angebunden
- Datenmodell und Datenbank-Anbindung (GORM) implementiert
- CI/CD-Pipeline für Deployment auf Docker-Basis erstellt
- OAuth2-Flow mit GitLab als Provider integriert
- Backend-Web-Framework: GoFiber
- Repository-Architektur zur flexiblen GitLab-Anbindung eingeführt
- API-Client: Go-Paket xanzy/go-gitlab angebunden
- Grundfunktionen wie Klassenraum- und Aufgaben-Erstellung implementiert
- Session-Management initial implementiert

08.11.23 - 4.02.2024

- Sprint 3: Backend-API für Klassenraum-Management erstellt
- Studierende per E-Mail zu Klassenräumen einladen und mit GitLab-Account beitreten
- GitLab-Gruppe für Klassenraum erstellt, Klassenraum verwaltbar im Frontend & Backend
- Einladungslink-Feature im Frontend integriert

- Benutzerverwaltung nach OAuth-Authentifizierung abgeschlossen
- E-Mail-Einrichtung, Frontend-Projektstruktur und Tailwind-Anpassungen vorgenommen
- Erste Frontend-Komponenten mit React-Bibliothek shadcn/ui erstellt
- Basis für Weiterentwicklung geschaffen

Sprint 4

Team-basierte Klassenräume

- Backend-API überarbeitet, um Aufgabenbearbeitung in Teams zu ermöglichen
- Große Backend-Veränderung: GitLab-Gruppen repräsentieren Klassenräume
- Untergruppen für Teams und Studierende, die Projekte pro Aufgabe beinhalten
- Frontend erweitert: Anzeige von Avataren und Benutzer-Logout-Funktion

Sprints 4–6

& UI-Überarbeitung

05.04.24 - 18.04.24

- Navigation und Avatar-URL in Benutzerabfragen integriert
- Test-Setup für CI/CD-Pipeline eingerichtet
- API-Endpunkte für Klassenraum-Bearbeitung hinzugefügt

Sprint 5 Feedback & Berechtigung

19.04.24 - 03.05.24

- Webhooks für GitLab-Gruppen untersucht, jedoch in der Community-Edition nicht verfügbar
- Entscheidung: Synchronisierungssystem zur Reaktion auf Änderungen entwickeln
- Erste Bewertungsfeatures implementiert: Feedback-Banches mit Merge Requests
- Konzepte für automatische Bewertungen und Tests geprüft
- Frontend-Komponenten für Aufgaben-, Mitglieder- und Teamliste aktualisiert
- Go-Paket swaggo/swag eingeführt, Swagger-Kommentare hinzugefügt
- Generative Dokumentation & API-Client-Code-Generierung mit OpenAPI-Spezifikation

Sprint 6 Frontend & Studierende

03.05.24 - 16.05.24

- Berechtigungen für Backend-API und GitLab-Gruppen (Klassenräume) überarbeitet

- Berechtigungen für Klassenraum-Mitglieder weiter implementiert
- Ersteller des Klassenraums hat Owner-Rechte, Studierende als Gäste mit Reporter-Rechten
- Studierende haben Maintainer-Rechte in Projekt-Repositories bis zur Fälligkeit der Aufgabe
- Moderatoren besitzen Reporter-Rechte im gesamten Klassenraum
- Frontend aktualisiert: "Klassenraum beitreten" und "Aufgabe annehmen"
- Avatar-Dropdown und Pop-up für Zuweisung von Klassenraum-Mitgliedern verbessert
- Backend für "Klassenraum beitreten" und "Aufgabe annehmen" überarbeitet

Sprints 7–10

Sprint 7 Fr & S 17.05.24 - 30.05.24

- Neue Layout im Frontend angepasst
- Neue Funktionalität: „Aufgaben bearbeiten“
- Namensgebung der Aufgaben angepasst
- Fehler Behoben

Sprint 8 Grading und UI

31.05.24 - 14.06.24

- Frontend an die überarbeitete Backend-API (aus Sprint 4) angepasst
- "Klassenraum erstellen"-Formular überarbeitet
- Funktionen zur Auswertung von Testergebnissen aus CI/CD-Pipelines im Backend implementiert
- Erster Schritt zur automatischen, testgetriebenen Bewertung
- Frontend-Routen an API-Version 2 angepasst
- Archivierungsfunktion für Klassenräume hinzugefügt

Sprint 9 Grading und UI II

14.06.24 - 28.06.24

- Login-Seite überarbeitet
- Automatisches Grading, Tests und Endpoints weiter implementiert
- Endpunkt zum Zurückziehen von Einladungen hinzugefügt
- Minimale Corporate Identity erstellt
- Frontend responsiver gemacht, einschließlich mobiler Ansichten für Menü und Benutzer-Menü
- Worker-System auf Go-Routinen entwickelt
- für Hintergrundaufgaben und GitLab-Synchronisierung

Sprint 10 Grading und UI III

- Aktualisierung der Klassenraum-Liste im Frontend zur Anpassung an Backend-Änderungen
- Einführung neuer Endpunkte zum Abrufen von Git-Repository-Links für Projektaufgaben
- Nutzung der Git-Repository-Links im Frontend für schnellen Zugriff auf Studierenden-Repositories

Sprints 10–12

27.06.24 - 19.07.24

- Identifikation fehlender Seiten und Erstellung entsprechender Issues
- Erstmals abgeschlossene Implementierung der Synchronisation mit dem Worker-System

Sprint 11 Frontend & Frontend

- Weiterentwicklung des Figma-Prototyps
- Vollständige Implementierung der Backend-Strukturen für die Bewertung
- Hinzufügung von Team-Slots im Teammanagement im Frontend
- Ergänzung eines Workers, der die Berechtigungen für fällige Aufgaben ändert, sodass Studierende keine Änderungen am Quellcode vornehmen können
- UI-Konzepts für die Konfiguration & Bewertung
- Implementierung des Exports von Bewertungsergebnissen in eine CSV-Datei
- Aktualisierung der visuellen Elemente beim Beitreten eines Klassenraums und beim Akzeptieren von Aufgaben

19.07.24 - 23.08.24

- Hinzufügung neuer Seiten wie: „Klassenräume bearbeiten“ & „Klassenraum-Übersicht“
- Überprüfung der Synchronisierung von Klassenraum- und Teamdaten
- Optimierung des Mail-Versands
- Ergänzung eines neuen Endpunkts für aktive Aufgaben

Sprint 12 Frontend & Dokumentation

24.08.24 - 13.09.24

- Ergänzung der fehlenden Endpunkte im Backend für Klassenraum-Mitglieder
- Aktualisierung der Ansicht der Aufgaben-Details im Frontend
- Hinzufügung neuer Seiten für Klassenraum-Mitglieder sowie Einstellungen für Klassenräume und Aufgaben
- Funktion der Bearbeitung von Aufgaben
- Synchronisations-Worker eröffnen nun geschlossene Aufgaben bei zukünftigen Fälligkeitsdaten
- Vollständige Überarbeitung der Integrationstests im Backend
- Frontend-Komponenten für Aufgabenliste und die Klassenraum-Ansicht für Stud.
- Strukturierung der Softwaredokumentation

Sprint 13

Sprint 13 Abgabe 06.09.24 - 27.09.24

- **Testung auf Fehler**
- **Entscheidungen über weitere Funktionalitäten wurden getroffen für Version 1**
- **Weiterarbeit an Dokumentation**
- **Bespr. über Veröffentlichung im OpenSource Bereich**

Technische Herausforderungen

Konzeptionierung einer Klassenraumstruktur in GitLab

Herausforderung und Ziel:

- 🐙 Umsetzung der Rechte und Strukturen von Klassenräumen in GitLab ohne Eingriff in den GitLab-Code; mehrere Ansätze wurden ausprobiert

Ergebnis

- 🐙 Eine geeignete Lösung wurde entwickelt, die die Verwaltung von Klassenräumen in einer beliebigen GitLab-Instanzen ermöglicht, ohne dass dafür Administrationsrechte erforderlich sind.

Technische Herausforderungen

Unerwartetes Verhalten der GitLab-API

- 🐙 GitLab-API zeigte unvorhersehbares Verhalten beim Forking von Projekten, da der Prozess asynchron abläuft.
- 🐙 Asynchrones Monitoring über 5 Minuten wurde implementiert, um den Abschluss des Fork-Vorgangs sicherzustellen und Branch-Schutzregeln korrekt anzuwenden.

Technische Herausforderungen

Verwendung von GORM zur objektrelationalen Abbildung

- 🐙 GORM verursachte technische Probleme, wie unvollständige Assoziationen und fehlende Unterstützung für automatisches Preloading, was manuelle Ergänzungen erforderte.
- 🐙 Herausforderungen bei Auto-Migration und fehlende Unit-Test-Möglichkeiten führten zur Entwicklung eines eigenen Migrationstools und zur Umstellung auf das Tool Goose.



Technische Herausforderungen

Vollständige Überarbeitung der Backend-API

- 🐙 Die Backend-API wurde zweimal überarbeitet: die erste Version bot nur grundlegende Funktionen, die zweite war zu komplex wegen separater Routen für erstellte und beigetretene Klassenräume.
- 🐙 In der finalen Version wurden Endpunkte vereinheitlicht, Routen zusammengeführt und Middlewares für kontext-bezogene Berechtigungsprüfungen eingeführt.

Technische Herausforderungen

Universelle Lösung für automatische Bewertung

-  Verschiedene Ansätze zur Ausführung und Auswertung von Tests wurden geprüft, wobei das GitLab-Feature "Unit test report" als die geeignetste Methode identifiziert wurde.
-  Diese Funktion ermöglicht die standardisierte Darstellung und Auswertung von Testergebnissen in CI/CD-Pipelines, wodurch Lehrende eine klare Übersicht erhalten und "GitClassrooms" die Ergebnisse zur automatischen Bewertung nutzen kann.



Technische Herausforderungen

Synchronisation mit GitLab

- 🐙 Die fehlenden Gruppen-Webhooks in der Open-Source-Version von GitLab erschwerten die Synchronisation zwischen der Anwendung und GitLab.
- 🐙 Als Lösung wurde ein Polling-basierter Ansatz implementiert, um Änderungen seitens GitLab an Klassenräumen, Projekten, Teams oder Benutzern zu erkennen.

Teambezogene Herausforderungen

Verschiedenes Vorwissen der Entwickler

-  Unterschiedliche Vorkenntnisse und technische Hintergründe der Entwickler erschwerten die Wahl eines gemeinsamen Tech-Stacks, wobei Go für das Backend und React für das Frontend ausgewählt wurden.
-  Um Wissenslücken zu schließen, wurden intensives Selbststudium und Pair Programming eingesetzt, wobei die zeitgleiche Teilnahme an einer Cloud-Engineering-Vorlesung den Lernprozess unterstützte.

Teambezogene Herausforderungen

Synchronisation zwischen Frontend und Backend

- 🐙 Die Synchronisation zwischen Frontend und Backend war problematisch, da das Backend schneller neue Features entwickelte als das Frontend, was zu Verzögerungen bei der UX-Überprüfung führte.
- 🐙 Es war wiederholt notwendig, Anpassungen an beiden Systemen vorzunehmen, um sicherzustellen, dass das Frontend nur die benötigten Daten erhält und verarbeitet.

Teambezogene Herausforderungen

Unterschiedlich starkes Engagement

- 🐙 Die wöchentlichen Meetings ermöglichten allen Teammitgliedern, zum Projektfortschritt beizutragen, wobei einige Entwickler durch überdurchschnittliches Engagement und proaktive Initiativen bei der Projektarchitektur und der Weiterentwicklung der Anwendung hervorstachen.









Vorstellung

Ausblick

- ☁ **Digitales Benutzerhandbuch**
- ☁ **Website zur Vorstellung von »GitClassrooms«**
- ☁ **Erweiterung der Tests**
- ☁ **Verbesserung der Code-Qualität**
- ☁ **Anbindung an Lernmanagement-Systeme**
- ☁ **Helm Chart**
- ☁ **Einbettung von Sentry**

Ausblick

-  **Einführung von Routen zum Löschen von Ressourcen**
-  **Anbindung an weitere Git-Lösungen wie Gitea und GitHub**
-  **Verbesserung der Synchronisation mit Webhooks**
-  **Überarbeitetes Logging**
-  **Überarbeitung der Authentifizierung**
-  **Eigene CLI zur Automatisierung von Prozessen**