# Softwaredesign Dokumentation

Software Engineering II

*Optima Connect*

**Projekttitel: BonoboBoard**

Jakob Hutschenreiter (1419081)
Jiesen Wang (9839152)
Nick Kramer (3122448)
Patrick Küsters (2598689)
Peter Moritz Hinkel (2783930)

DHBW Mannheim
26. Februar 2022

## Änderungshistorie

| Revision | Datum | Autor(en) | Beschreibung |
|---|---|---|---|
| 1.0 | 24.01.2022 | JW | A: 1, 3 |
| 1.1 | 25.01.2022 | JW\|PH\|NK\|PK | A: 2, 4, 5 |
| 1.2 | 26.01.2022 | JH | A: 4, 6 |

Abkürzungen: Hinzugefügt/Added (A), Änderung/Changed (C), Löschung/Deleted (D)

## Inhaltsverzeichnis

# 1 Motivation

Um sicher zu gehen, dass alle Softwaredesignziele des BonoboBoard-Projekts richtig umgesetzt werden, werden diese in diesem Dokument festgehalten. Das Dokument erleichtert die Analyse, Planung, Implementierung und Entscheidungsfindung. Anders als im Anforderungsdokument wird nun hier beschrieben wie die Ziele umgesetzt werden.

# 2 Entwicklersetup

Verwendete Software und Libraries Versionen:

| **Software/Library** | **Version** |
|---|---|
| Django | 4.0.2 |
| Django Tables | 2.4.1 |
| dj-database-url | 0.5.0 |
| Gunicorn | 20.1.0 |
| Docker | 20.10.12, Build e91ed57 |
| PyCharm | 2021.3.2 |
| VisualStudio Code | 1.64.2 |
| Beautiful Soup | 4.8.2 |
| Requests | 2.27.1 |
| iCalendar | 4.0.9 |
| Pandas | 1.4.0 |
| lxml | 4.5.0 |
| SQLAlchemy | 1.4.31 |
| asgiref | 3.4.1 |
| autopep8 | 1.6.0 |
| Python | 3.9 |
| pycodestyle | 2.8.0 |
| python-decouple | 3.5 |
| pytz | 2021.3 |
| sqlparse | 0.4.2 |
| toml | 0.10.2 |
| Unipath | 1.1 |
| whitenoise | 5.3.0 |

## 2.1 Hosting bei 1Blu

Um die Webanwendung anderen Studierenden zur Verfügung stellen zu können, haben wir uns dafür entschieden, die Anwendung auf einem virtuellen privaten Server (VPS) laufen zu lassen. Der VPS läuft bei dem Hosting-Anbieter 1Blu. Da die Anwendung mit Docker bereitgestellt wird, hält sich der Konfigurationsaufwand auf dem Server in Grenzen. Auch ein einfaches Neu-Bauen ist möglich, was im automatisierten Deployment-Prozess genutzt wird (siehe Abschnitt 4.3).

# 3 Verwendete Tools

Für ein gemeinsames strukturiertes Arbeiten am Projekt, werden je nach Aufgabe verschiedene Tools eingesetzt.

## 3.1 Tools für das Projektmanagement/Organisation

| Tool | Einsatzzweck |
|---|---|
| Discord | Austausch von Nachrichten, Virtuelle Meetings, Umfragen, Informationsmanagement |
| WhatsApp | Austausch von Nachrichten |
| DropBox | Informationsmanagement |
| Google Kalender | Terminplanung |
| Jira | Aufgaben- und Projektmanagement, Prozessmanagement |
| Latex und Git | Dokumente erstellen und bearbeiten |
| OneNote | Informationsmanagement, Dokumentvorlagen erstellen und bearbeiten |
| Draw.io | Diagramme erstellen und bearbeiten |
| Git und GitHub | Versionskontrolle und Repository-Verwaltung |
| Adobe XD | Ideen Entwicklung, Erstellung von Mockups |

## 3.2 Tools für die Softwareentwicklung

Diese sind dem separaten Dokument: „Tools für die Softwareentwicklung“ zu entnehmen.

# 4 Zentrale Designentscheidung

Im folgenden werden die zentralen Designentscheidungen aufgegriffen und erläutert.

## 4.1 Backend: SOLID-Prinzip

Für die Interaktion mit den DHBW Webseiten bietet sich das SOLID-Prinzip an.
Gründe hierfür sind:

- Die Zugriffsmöglichkeiten auf die benötigten Inhalte (DHBW-Webseiten) sind Subjekt für Änderungen, daher Abstraktion dieser Inhalte (mittels Klassen und Klassenattributen), sodass das Frontend weiterhin die gleichen Schnittstellen benutzen kann.
- Die gleichen Schnittstellen wie für das Frontend erleichtern auch das standardisierte Persistieren der Daten in eine Datenbank.
- Zur Interaktion braucht es immer HTTP Anfragen, somit bietet sich eine Oberklasse A an, welche diese Logik implementiert und den Unterklassen einfach anbietet.
- Bei den Webseiten: Zimbra, Dualis und Moodle braucht es eine Authentifizierung des Endnutzers, folglich ist eine weitere Oberklasse B, welche die Oberklasse A um die Funktionalität zum Authentifizieren erweitert, vonnöten.

- Die Interaktion mit Dualis, Moodle und dem Vorlesungsplan hat als Ziel den Erhalt und die Filterung der Daten; Zimbra hat allerdings den Erhalt / die Filterung der Daten und das Verschicken der Daten als Funktionalität. Hier bietet sich es ebenfalls an, zwei Oberklassen einzuführen, welche die eben genannten Punkte unterstützen.

### 4.2 Frontend: DRY-Prinzip

Im Frontend haben wir Zugriff auf die abstrakte Darstellung der "Datensätze" aus dem Backend und die vorgegebenen Funktionen / Klassen von Django. Die einzige Aufgabe besteht darin, die Daten zu visualisieren und den entsprechenden Pfaden zu zuordnen. Mit diesen Gegebenheiten wird im Frontend "nur" darauf geachtet, dass keine Code-Duplizierungen auftreten, wie es das DRY-Prinzip vorgibt.

### 4.3 Automatisierter Workflow (CI-Pipeline)

Damit der Test und Deployment-Prozess nicht manuell ausgeführt werden muss, nutzen wir GitHub-Actions zur Verwaltung der Continous Integration (CI). Dabei nutzen wir zwei Workflows:

- Ausführen der Unittests bei jeder Eröffnung eines Pull-Request
- Ausführen der Unittests und automatisches Deployment auf den VPS bei Änderungen auf dem Main-Branch

Da das automatische Deployment eine komplexere Schrittfolge beinhaltet, beschreiben wir diese genauer. Über GitHub-Actions wird ein Ordner erzeugt, welcher alle Dateien beinhaltet, um die Produktivumgebung auf dem Server zu starten. Dieses Artefakt wird auf den Server kopiert und anstelle der alten Produktivinstanz eingesetzt. Vorher ist es notwendig, die SSL-Zertifikate zu sichern und nach dem Umstellen wieder einzusetzen.

## 5 UML

Das gesamte UML ist dem Link: BonoboBoard UML zu entnehmen.

## 6 Zusammenfassung

Das BonoboBoard wird sich im Frontend am DRY-Prinzip orientieren und im Backend an dem SOLID-Prinzip. Eine erste prototypische Version dieser Orientierung ist im UML-Diagramm dargestellt. Das Projekt wird auf einem VPS bei 1Blu gehostet und mit Docker "containerized". Auf dieser "Containerization" baut die CI-Pipeline auf, für ein automatisches Testing und Deployment.