



Wackeldisplay Wetter

Projektdokumentation



Abbildung 1 Foto von Pedro Mirano auf Unsplash

89grad GmbH

Yusup Khasbulatov

Projektdokumentation

Wackeldisplay Wetter

Projektangaben gemäss Pkorg

Firmenname	89grad GmbH
Abteilung	IT
Berufsschule	gibb
Valid-Experte	Lawson Mike
Hauptexperte	Engler Hans
Nebenexperte	Georg Achermann
Berufsbildner	Ramun Hofmann
Fachrichtung	Applikationsentwicklung
Projektvorgehensmodell	HERMES-gibb
Jahrgang	2022
Ausgabedatum	23.05.2022

Tabelle 1 Projektangaben gemäss Pkorg

Kurzfassung des IPA Berichtes

Ausgangssituation

Im Moment haben wir ein funktionierendes "Wackeldisplay", das erfolgreich die Abfahrtszeiten der öffentlichen Verkehrsmittel von den beiden dem Büro am nächsten gelegenen Haltestellen anzeigt. Die fünf beweglichen Spalten zeigen verschiedene Symbole, die die eine oder andere Information darstellen. Die Daten für diese Anzeige werden von einer öffentlichen Datenquelle <https://opendata.ch> erhalten. Durch den Aufruf von URL <https://transport.opendata.ch/v1/stationboard?id=8571393&limit=30> wird eine JSON-Zeichenkette zurückgegeben. Diese Zeichenkette wird überarbeitet, indem unnötige Daten ausgefiltert werden. Danach wird sie an eine Schleife übergeben, die die Daten durchläuft und die Steuerungsmethode von Spalten aufruft.

Diese Methoden ihrerseits senden über das Netzwerk unter Verwendung des MQTT-Protokolls Anfragen an die entsprechende MAC-Adresse des ESP8266-Mikrocontrollers. Diese Anfragen enthalten die Positionen, zu denen die Schrittmotoren die Säulen verschieben sollen.

Umsetzung

Die Software zur Steuerung des Displays wird leicht verändert. Die Möglichkeit, die Abfahrtszeiten der öffentlichen Verkehrsmittel anzuzeigen, wird durch eine Anzeige des Wetters in 5 Orten und den Trend für die nächsten 3-5 Tage ersetzt. Für die Umsetzung wird eine öffentliche Datenquelle verwendet, die nach der Auswertung in der Initialisierungsphase des Projektes nach dem Variantenvergleich und der Variantenentscheidung festgelegt wird.

Ergebnis

Die User Experience des Displays soll durch die Erhöhung der Geschwindigkeit des Schrittmotors verbessert werden (dieser Prozess wird mit der engen Unterstützung von Herrn Florian Baumgartner durchgeführt, da die Software auf ESP8266 von ihm entwickelt wurde und in C++ geschrieben ist). Darüber hinaus soll ein Web-Interface zur Steuerung des Wackeldisplays entwickelt werden, genauer gesagt zur Anzeige des Wetters für den gewählten Tag.

Inhaltverzeichnis

Inhaltverzeichnis.....	3
Teil 1: Ablauf und Umfeld	7
1. Aufgabenstellung.....	8
1.1. Titel der Arbeit	8
1.2. Ausgangslage	8
1.3. Detaillierte Aufgabenstellung	8
1.4. Mittel und Methoden.....	8
1.5. Vorkenntnisse	9
1.6. Vorarbeiten	9
1.7. Neue Lerninhalte.....	9
1.8. Arbeiten in den letzten 6 Monaten.....	9
2. Standards	10
3. IPA-Schutzbedarfsanalyse.....	11
3.1. Zugriff auf lokale Computer.....	11
3.2. Zugriff auf GitLab.....	11
3.3. Zugriff auf Django Admin Bereich.....	11
3.4. Zugriff auf den Wackeldisplay Server.....	11
3.5. Zugriff auf ESP8266 Microcontroller (Software Patch)	11
3.6. Zugriff auf NAS (IPA Daten-Backup).....	12
3.7. Zugriff auf E-Mail	12
4. Organisation der IPA Ergebnisse.....	13
4.1. Arbeitsumgebung.....	13
4.2. Datensicherung der IPA.....	15
5. Projektvorgehen	20
5.1. Projektvorgehensmodell	21
5.2. Szenario	21
5.3. Phasen	21
5.4. Module	22
5.5. Meilensteine.....	23
6. IPA Projektorganisation inkl. Projektrollen.....	24
6.1. Organigramm.....	24
6.2. Projektrollen.....	24
7. Zeitplan	26

8. Arbeitsjournale	27
8.1. Tag 1: 02.05.2022.....	27
8.2. Tag 2: 05.05.2022.....	29
8.3. Tag 3: 06.05.2022.....	31
8.4. Tag 4: 09.05.2022.....	33
8.5. Tag 5: 12.05.2022.....	34
8.6. Tag 6: 13.05.2022.....	36
8.7. Tag 7: 16.05.2022.....	38
8.8. Tag 8: 19.05.2022.....	40
8.9. Tag 9: 20.05.2022.....	41
8.10. Tag 10: 23.05.2022	43
9. Abschlussbericht	45
9.1. Vergleich Ist/Soll	45
9.2. Persönliches Fazit.....	45
9.3. Schlussreflexion.....	45
Teil 2: Projektdokumentation	46
10. Einleitung.....	47
11. Initialisierung	49
11.1. Studie; Ist-Zustand	49
11.2. Stärken der IST-Situation	68
11.3. Schwächen der IST-Situation.....	68
11.4. Persönliche Vorgehensziele.....	69
11.5. Projektziele.....	69
11.6. Grenzwerte.....	70
11.7. Abgrenzungen	70
11.8. Anforderungen	70
11.9. Risikoanalyse.....	72
11.10. Risikograph	73
11.11. Lösungsvarianten.....	73
11.12. Variantenantrag mit Begründung.....	75
12. Konzept.....	76
12.1. Systemanforderungen.....	77
12.2. Benutzerschnittstelle und Designkonzept	85
12.3. Materialbeschaffung.....	87
12.4. Systemarchitektur	88

12.5. Testkonzept.....	88
13. Realisierung.....	92
13.1. Wackeldisplay Hardwareanpassung	92
13.2. Projekt Struktur	93
13.3. Automatisation und Migration Skripte	94
13.4. MVC (Django MTV) Komponente.....	96
13.5. Routing.....	97
13.6. Datenverwaltung (Django-Admin)	100
13.7. Wackeldisplay Worker (Python Skript)	104
14. Einführung.....	106
14.1. Installation (Deployment).....	106
14.2. Benutzeranleitung.....	106
15. Abbildungsverzeichnis.....	107
16. Tabellenverzeichnis	108
17. Literatur- und Quellenverzeichnis	111
18. Abkürzungverzeichnis	115
19. Glossar	116
20. Anhänge.....	119
20.1. Phasen Freigabe	119
20.2. Sitzungsprotokolle	121
20.3. Coding Convention vom 89grad GmbH	130
20.4. Design Guidelines vom 89grad GmbH.....	131
20.5. Testprotokoll.....	132
20.6. Dokumentvorlage 89grad GmbH.....	135
20.7. Versionsverwaltung mit Verwaltungssoftware	135
20.8. Ordnerstruktur und Ablage der Dokumentationen.....	140
20.9. Projekt Code.....	141
20.9.1. README Datei.....	141
20.9.2. Wackeldisplay Worker (Python Skripte).....	142
20.9.3. Automatisation und Migration Skripte	149
20.9.4. Skripte für die Recherche und das Parsing von Daten.....	155
20.9.5. Routing	159
20.9.6. Django Admin Einstellung.....	159
20.9.7. Modelle.....	161
20.9.8. Templates (Views)	163

20.9.9. Views (Controllers)	167
20.9.10. Hilfeskripte.....	168
20.9.11. Django Settings	170
20.9.12. Tests.....	174
20.9.13. Testdaten	180
Benutzerhandbuch.....	193

Teil 1: Ablauf und Umfeld

Projektname: Wackeldisplay Wetter

Autor: Yusup Khasbulatov

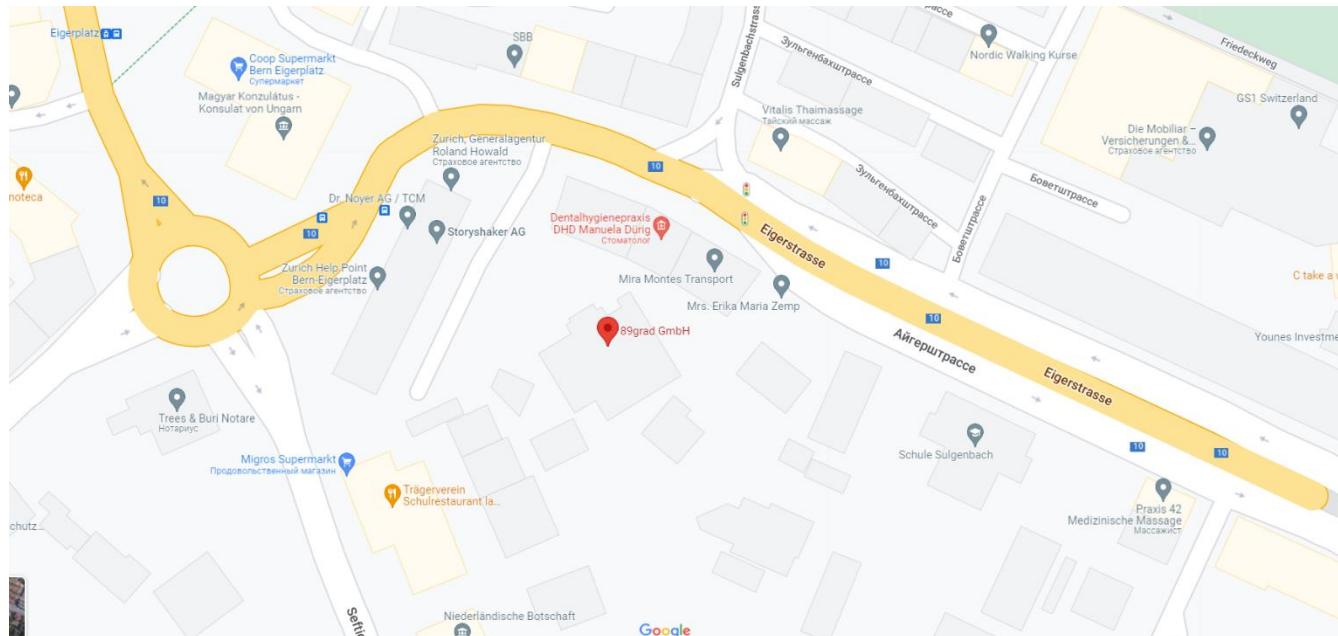


Abbildung 2 89grad GmbH auf der Karte

89grad GmbH

Eigerstrasse 12, 3007 Bern

info@89grad.ch

+41 31 511 85 89

1. Aufgabenstellung

1.1. Titel der Arbeit

Erstellen einer Komponente eines Web-Tools (Erweiterung Wackeldisplay)

1.2. Ausgangslage

89grad hat zur Visualisierung von Daten ein IoT-Hardware-Display gebaut. Dieses besteht aus 5 verschiedenen Segmenten, die jeweils Daten anzeigen. Darauf werden aktuell Fahrplan-Daten angezeigt und man sieht, wann der nächste Bus (Segment 1: Bus/Tram/Zug) ab Eigerplatz (Segment 2: Haltestelle) in welche Richtung (Segment 3) auf welchem Gleis (Segment 4) in wieviel Minuten (Segment 5) fährt.

Jedes Segment (transparente Kunststoffplatte, die von einem Schrittmotor angetrieben wird) kann bei Bedarf durch ein längeres oder kürzeres Segment ersetzt werden. Diese Segmente sind auch mit Symbolen als Datenanzeige versehen, z. B. Symbole für Bus, Tram, Zug, Haltestelle, Abfahrtszeit usw. Diese Symbole können bei Bedarf durch andere Symbole ersetzt werden.

1.3. Detaillierte Aufgabenstellung

Im Rahmen der IPA muss der bestehende Prototyp geändert werden, um eine andere Datenquelle anzuzeigen. Statt des Fahrplans sollen in Zukunft Wetterdaten an verschiedenen Orten angezeigt werden.

1. Es muss eine Auswertung der verschiedenen Wetterquellen (API) im Internet erstellt werden. Die Kriterien für den Vergleich sollten zusammen mit dem PO entwickelt werden. Auf dieser Grundlage muss die Wetterdatenquelle ausgewählt werden.
2. Das Backend muss an die neue Datenquelle angepasst werden.
3. Das System muss so konfiguriert werden (Hardware und Software), dass es Wetterdaten auf dem Wackeldisplay anzeigen können. Das Display zeigt 5 Segmente an: Ort, Temperatur, Wolken/Sonnenschein, Niederschlag (mm), Trend für die nächsten 3 Tage.
4. Die resultierenden Daten (Ort, Temperatur, Bewölkung/Sonnenschein, Niederschlag (mm), Trend für die nächsten 3 Tage) müssen als Tabelle auf einer Webseite dargestellt werden.
5. Es muss das Wetter für folgende Orten auf dem Display angezeigt werden: Bern, Zürich, Paris, London, San Francisco (inkl. dazugehöriger Daten).
6. Es muss möglich sein, bestimmte Zeilen aus der Tabelle auf dem Wackeldisplay anzuzeigen.
7. Es muss eine Benutzeranleitung für die Bedienung des Wackeldisplays erstellt werden.

Zur Versionsverwaltung soll Git eingesetzt werden. Der Kandidat verwendet hierzu den GitLab Server von 89grad. Darin sollen Daten, Code und die Dokumente verwaltet werden. Die gespeicherten Versionen müssen aussagekräftig angeschrieben sein.

1.4. Mittel und Methoden

- Entwicklung in Python/Django für das Backend und die API-Anbindung.
- Entwicklung in HTML/JS/Bootstrap (o.ä.) für das Frontend.
- Firmenvorgaben gemäss "Coding Guidelines" von 89grad (Quelle: wiki.89grad.ch)
- Projektmethode: HERMES-gibb gemäss Modul 306

1.5. Vorkenntnisse

- Konzeptionelle/Logische/Physische Datenbank Erstellung
- Diverse Projekt Mockups/Design/Prototyp Erstellung
- Projekt Tests (Unit Test, Integration Tests, E2E Tests)
- Projekt Dokumentation
- Problemlösungen in diversen Projekten (Python, HTML, JavaScript)

Der Lernende hat praktisch während der ganzen Lehrzeit mit diesen Produkten/Techniken gearbeitet.

1.6. Vorarbeiten

- Erstellung des Templates für die Dokumentation der Arbeit.
- Einrichtung des Arbeitsplatzes für die IPA.
- Lesen von alten 89grad IPAs
- (Wieder-)Einlesen in den bestehenden Wackeldisplay-Code

1.7. Neue Lerninhalte

Die Arbeit enthält keine komplett neuen Aspekte. Der Kandidat sollte über alle Grundlagen für die erfolgreiche Entwicklung der Arbeit verfügen. Neu wird die Kombination verschiedener bekannter Inhalte sein.

1.8. Arbeiten in den letzten 6 Monaten

Wackeldisplay

Das Wackeldisplay ist ein von der 89grad GmbH entwickeltes Gerät. Dieses Gerät wird durch IoT-Hardware gesteuert. Herr Khasbulatov migrierte und passte die Skripte in einer Django-basierten Webanwendung an. Die ursprünglich hart codierten Daten wurden durch Opendata.ch Fahrplan REST API-Abfragen ersetzt.

Zwei Wochen Aufwand wurden in diesen Auftrag investiert.

Smartaccess

Smartaccess ist eine in Zusammenarbeit mit der SBB entwickelte mobile Anwendung. Es ist grundsätzlich ein Verwaltungstool für SBB-Schliessfächer mit begrenztem Zugang. Es handelt sich um eine hybride Anwendung, die mit dem Ionic-Framework von Herrn Khasbulatov entwickelt wurde.

89grad Mobile App

Herr Khasbulatov hat für die 89grad GmbH eine Android-Applikation entwickelt, die als Informationsplattform über die 89grad GmbH dient. Die Anwendung wurde von Herrn Khasbulatov im Google Play Market unter dem Firmenkonto der 89grad GmbH veröffentlicht.

2. Standards

Standard	Beschreibung
Dokumentenvorlage	Offiziell verwendet 89grad GmbH üblicherweise die folgende Vorlage (siehe Dokumentenvorlage 89grad GmbH) für die Dokumente. Nach Absprache mit dem Fachvorgesetzten wurde jedoch die aktuelle Vorlage genehmigt.
Versionierung der SW	Die 89grad GmbH betreibt einen eigenen GitLab-Server und versioniert die Projekte auf dieser Plattform.
Code / Skripte / Kommentare	Siehe Coding Convention vom 89grad GmbH .
Design Guidelines	Siehe Design Guidelines vom 89grad GmbH .
Schriftart	89grad GmbH verwendet in ihren Projekten zwei Schriftarten: Arial und Open Sans.
Sicherheitskonzept	89grad GmbH verfügt über kein Sicherheitskonzept. Bei diesem Projekt gibt es keine sensiblen Daten, die geschützt werden müssen, aber es gibt einige Schnittstellen, bei denen Sicherheitselemente involviert sind; diese sind in der IPA-Schutzbedarfsanalyse beschrieben.
Dateiablage	Zur Datenspeicherung nutzt die 89grad GmbH einen eigenen Cloud-Speicher auf Basis der Open Source Plattform OwnCloud. Dieser wird auch für die Dateiablage dieses Projekts verwendet.
Projektmethode	Die 89grad GmbH hat keine Standards für die Projektmanagementmethode. Für dieses Projekt wird HERMES-gibb verwendet, welche vom Fachvorgesetzten akzeptiert wurde.

Tabelle 2 Projektstandards

3. IPA-Schutzbedarfsanalyse

Die folgenden Bereiche dieses Projekts sind geschützt und haben keinen oder nur teilweisen öffentlichen Zugang.

3.1. Zugriff auf lokale Computer

Der Arbeitslaptop ist nur nach Anmeldung nutzbar und durch ein Passwort geschützt.

3.2. Zugriff auf GitLab

Der Zugriff auf den GitLab-Server der 89grad GmbH erfordert einen aktivierten Benutzeraccount. Die verschiedenen Bereiche und Projekte in GitLab sind ebenfalls vor unerwarteten Zugriffen geschützt. Wenn man auf einen geschützten Bereich zugreifen möchte, muss man sich an die Operations-Abteilung wenden. Ein temporärer Account kann auch für einen externen Kunden angelegt werden, z.B. wenn die 89grad GmbH an einem Projekt mit einer externen Firma arbeitet.

3.3. Zugriff auf Django Admin Bereich

Die Wackeldisplay-Backend-Applikation verfügt über einen Administrationsbereich, der mit dem Django Web Framework erstellt wurde. Der Zugriff auf diesen Bereich ist nur möglich, wenn der User ein Benutzerkonto besitzt.

3.4. Zugriff auf den Wackeldisplay Server

Der SSH-Zugang wird über die Operations-Abteilung eingerichtet.

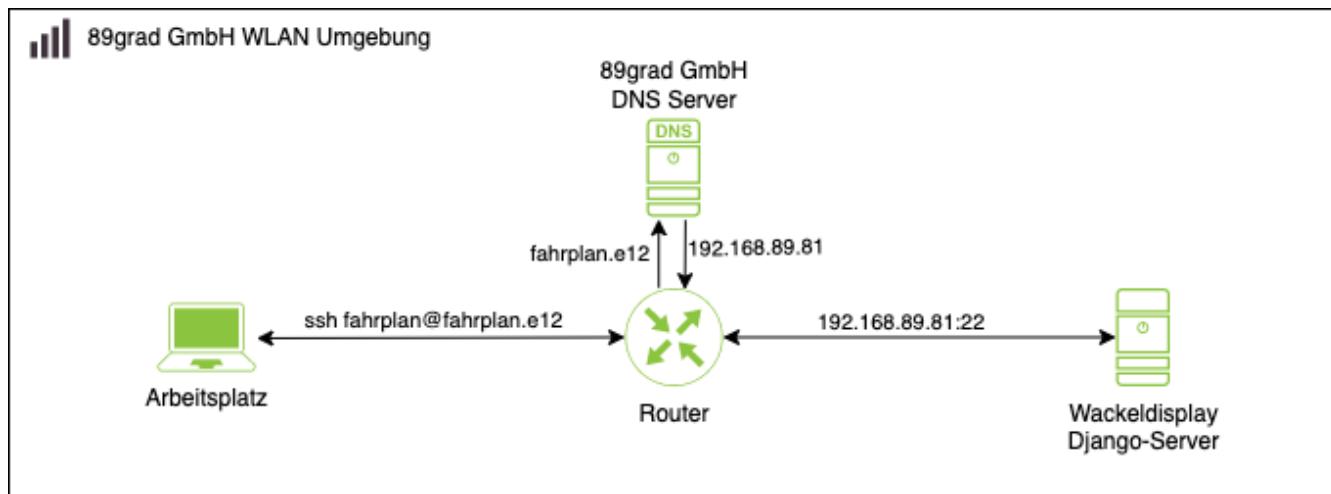


Abbildung 3 89grad GmbH DNS

3.5. Zugriff auf ESP8266 Microcontroller (Software Patch)

Die ESP8266-Mikrocontroller sind nicht geschützt und jeder kann sie praktisch herausnehmen und neue Software darauf installieren. Eine Verbesserung dieses Problems ist im Rahmen dieses Projekts jedoch nicht vorgesehen.

3.6. Zugriff auf NAS (IPA Daten-Backup)

Die 89grad GmbH verfügt über ein NAS, auf dem man seine Datenablage organisieren kann. Der Zugriff auf dieses NAS ist nur möglich, wenn ein persönliches Benutzerkonto vorhanden ist. Dieses NAS wird in diesem Projekt als Hauptgerät für die Datensicherung verwendet.

3.7. Zugriff auf E-Mail

Der Zugang zum E-Mail-Konto erfolgt über einen Benutzernamen und ein Passwort. E-Mails werden in praktisch allen oben genannten Softwareprodukten zum Zurücksetzen von Passwörtern verwendet.

4. Organisation der IPA Ergebnisse

4.1. Arbeitsumgebung

4.1.1. Softwareliste

Name	Beschreibung
MacOS Monterey	Betriebssystem
Chrome	Webbrowser
PyCharm	IDE für die Entwicklung auf Python
MS Office	Dokumenten-Bearbeitung
Slack	Kommunikation
Draw.io	Drawing Tool
DBeaver	Datenbank Management Applikation (Browser)
Adobe Produkte	Grafik, PDF, Vektor Grafik

Tabelle 3 Projekt Software Liste

4.1.2. Arbeitsplatz

Während der ganzen IPA wird am folgenden Arbeitsplatz gearbeitet.



Abbildung 4 Arbeitsplatz von vorne

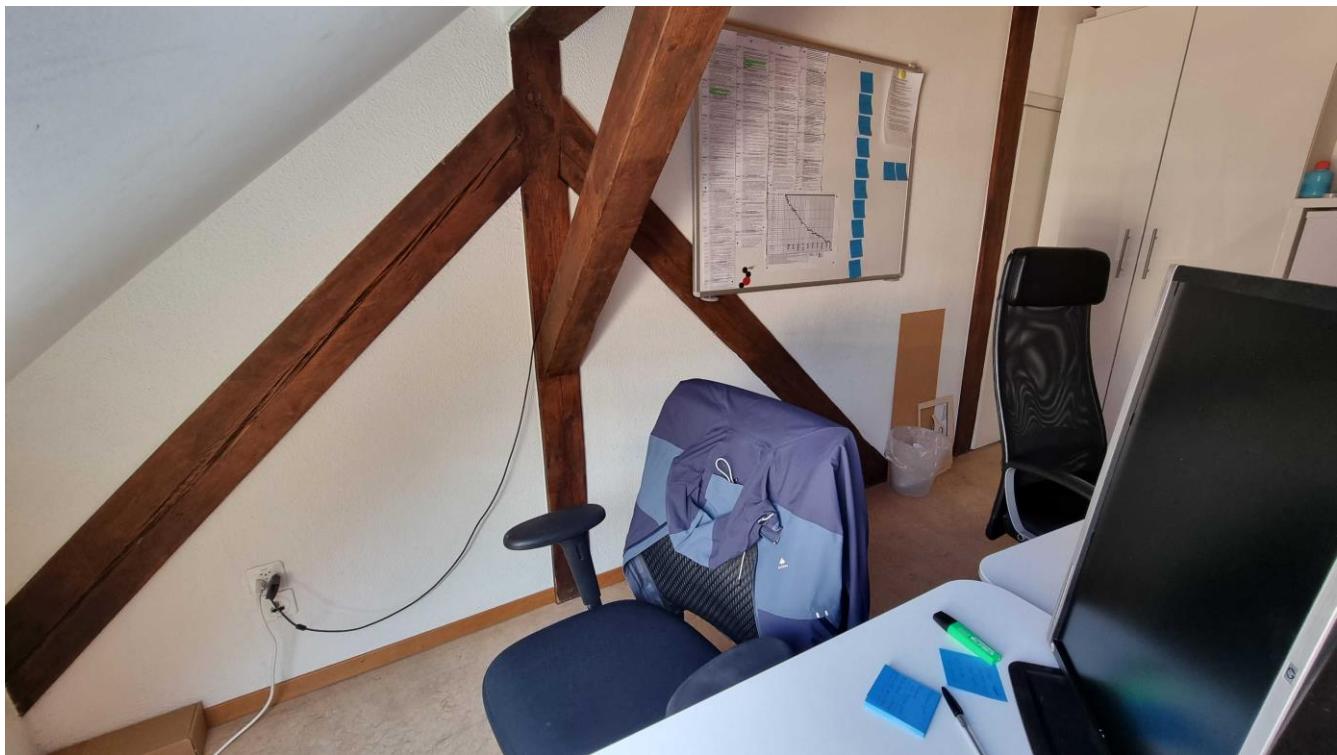


Abbildung 5 Arbeitsplatz von hinten

4.1.3. Laptop



Die Konfiguration meines Arbeitslaptops.

Abbildung 6 Laptop Konfiguration



Ersatz-Laptop. Wird verwendet, wenn der Haupt-Laptop nicht mehr funktioniert. Siehe [Risiken](#)

4.2. Datensicherung der IPA

Für die Ablage aller Daten wird, wie im Teil Firmenstandards beschrieben, die OwnCloud der 89grad GmbH verwendet.

4.2.1. Filestruktur der gespeicherten Daten

Die Filestruktur wurde, wie im Abschnitt Tipps und Tricks für die Dokumentation von Pkorg beschrieben, zweckmässig und übersichtlich aufgebaut.

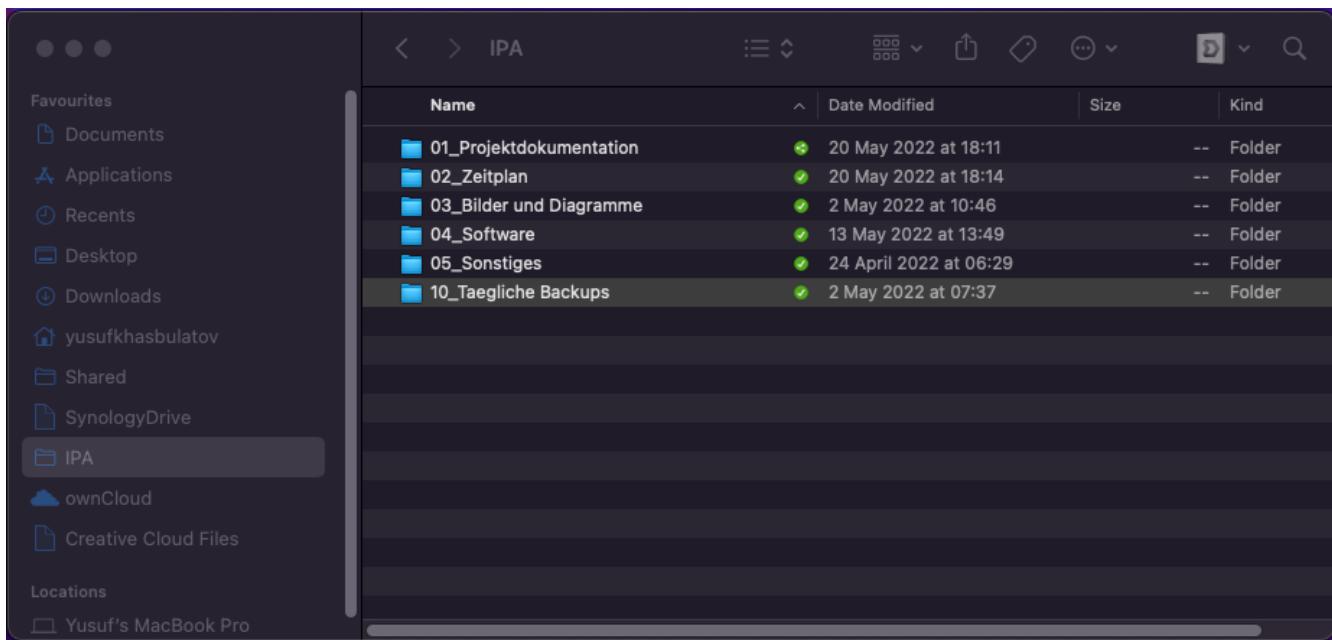


Abbildung 7 Filestruktur

Im Ordner "Taegliche Backups" werden die nummerierten Ordner zweimal täglich hineinkopiert. Der Ordner "Taegliche Backups" wird jeden Tag am Abend auf einen USB-Stick kopiert.

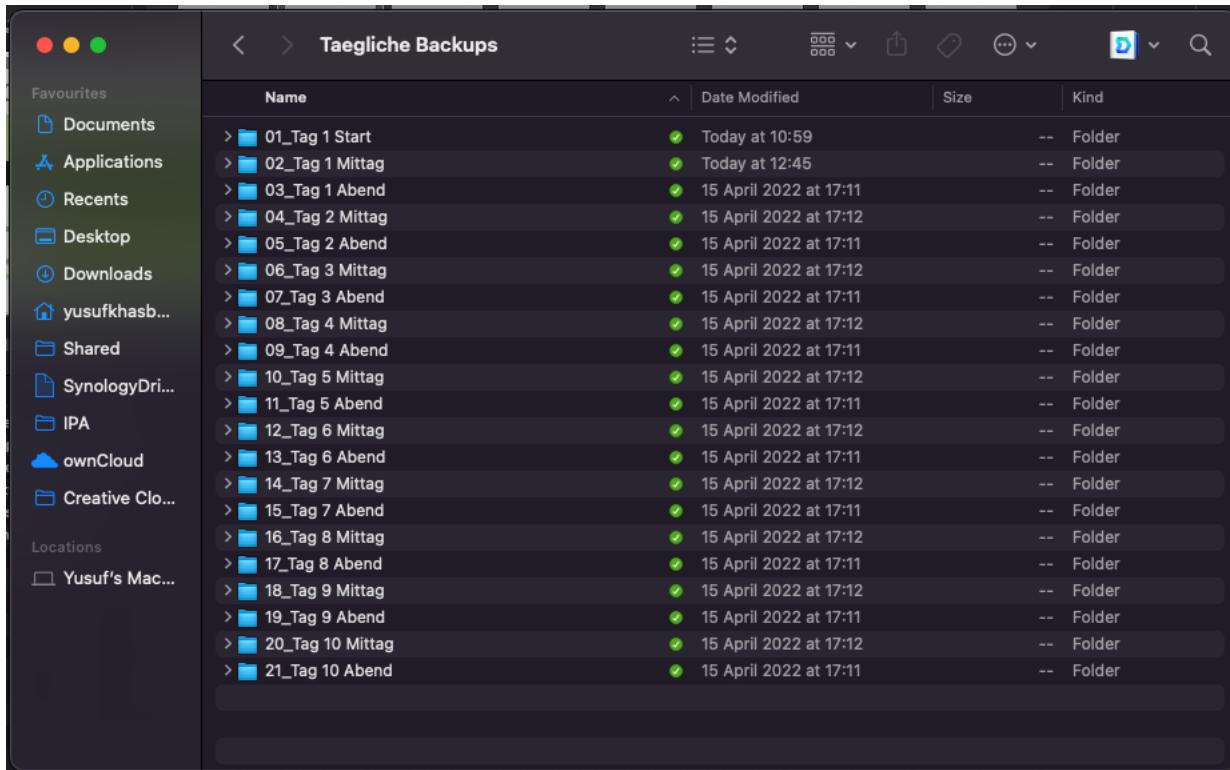


Abbildung 8 Tägliche Backups Ordner Struktur

4.2.2. Versionierung Dokumente

Die Dokumentation wird, wie im Kapitel [Detaillierte Aufgabenstellung](#) erwähnt, auf dem GitLab Server vom 89grad GmbH versioniert, wie man in der Abbildung oben sehen kann.

Yusuf Khasbulatov > IPA-Documentation

IPA-Documentation

Project ID: 590

-o 5 Commits 1 Branch 2 Tags 2.6 MB Files 2.6 MB Storage

main ipa-documentation / History Find file Web IDE Clone

docs(new content): Standards
Yusuf Khasbulatov authored 24 minutes ago 968050c6

Upload File Add README Add LICENSE Add CHANGELOG Add CONTRIBUTING Add Kubernetes cluster

Set up CI/CD Configure Integrations

Name	Last commit	Last update
.gitignore	chore(configuration) Gitignore file	3 hours ago
Projektdokumentation Vorlage....	feat(template): Documentation Template	3 hours ago
Projektdokumentation.docx	docs(new content): Standards	24 minutes ago

Abbildung 9 Übersicht der GitLab IPA-Dokumentation

Yusuf Khasbulatov > IPA-Documentation > Tags

Tags give the ability to mark specific points in history as being important

Filter by tag name Updated date New tag

Tag_8_Abend -o d59981af · docs(no new content): Neue Inhalt · 23 hours ago
Tag_7_Abend -o c0a02321 · docs(no new content): Arbeitsjournal · 3 days ago
Tag_6_Mittag -o 537e1c11 · docs(new content): Sitzungsprotokoll Expertenbesuch · 6 days ago
Tag_5_Mittag -o 1a85ee1e · docs(new content): Sitzungsprotokoll (Sitzung 6) · 1 week ago
Tag_4_Abend -o 0abd3f4e · docs(new content): Arbeitsjournal · 1 week ago
Tag_4_Mittag -o ca537e3b · docs(new content): Konzept Benutzerschnittstelle und Designkonzept · 1 week ago
Tag_3_Abend -o c4b6c54e · docs(new content): Arbeitsjournal · 1 week ago
Tag_3_Mittag -o 6fa59593 · docs(new content): Funktionale und nicht Funktionale Anforderungen · 1 week ago
Tag_2_Abend -o 45926938 · docs(new content): Arbeitsjournal 05.05.2022 · 2 weeks ago
Tag_2_Mittag -o 8577182a · docs(new content): Sitzungsprotokoll und Einleitung · 2 weeks ago
Tag_1_Abend -o 2c83992b · docs(new content): Arbeitsjournal und Sitzungsprotokoll · 2 weeks ago
Tag_1_Mittag -o b8bb8db1 · docs(new content): Kurzfassung des IPA Berichtes · 2 weeks ago
Tag_1_Start -o ef9a6a32 · chore(configuration) Gitignore file · 2 weeks ago

Abbildung 10 GitLab IPA-Dokumentation Tags

4.2.3. Wiederherstellung

Die Wiederherstellung der Daten ist von drei Stellen aus möglich:

USB-Stick

Die gewünschten Daten auswählen, kopieren und ersetzen.

OwnCloud

OwnCloud erstellt automatisch eine Version nach jeder Änderung des Dokuments. Die gewünschte Version kann zum Herunterladen ausgewählt werden.

The screenshot shows the OwnCloud web interface. On the left, a sidebar menu includes 'All files' (selected), 'Favorites', 'Shared with you', 'Shared with others', 'Shared by link', 'Tags', 'Deleted files', and 'Settings'. The main area displays a list of files under 'All files'. Two files are listed: 'Projektdokumentation.docx' (1.8 MB, 25 minutes ago) and 'Projektdokumentation Vorlage' (235 KB, 14 hours ago). Below the list, it says '2 files' and '2.1 MB'. To the right, there is a detailed view of the 'Projektdokumentation.docx' file, showing its version history. The 'Versions' tab is selected, displaying eight versions of the document, each with a download icon, timestamp, size, and a refresh/circular arrow icon. The timestamps range from '33 minutes ago' to '4 hours ago'.

Abbildung 11 OwnCloud Daten Version

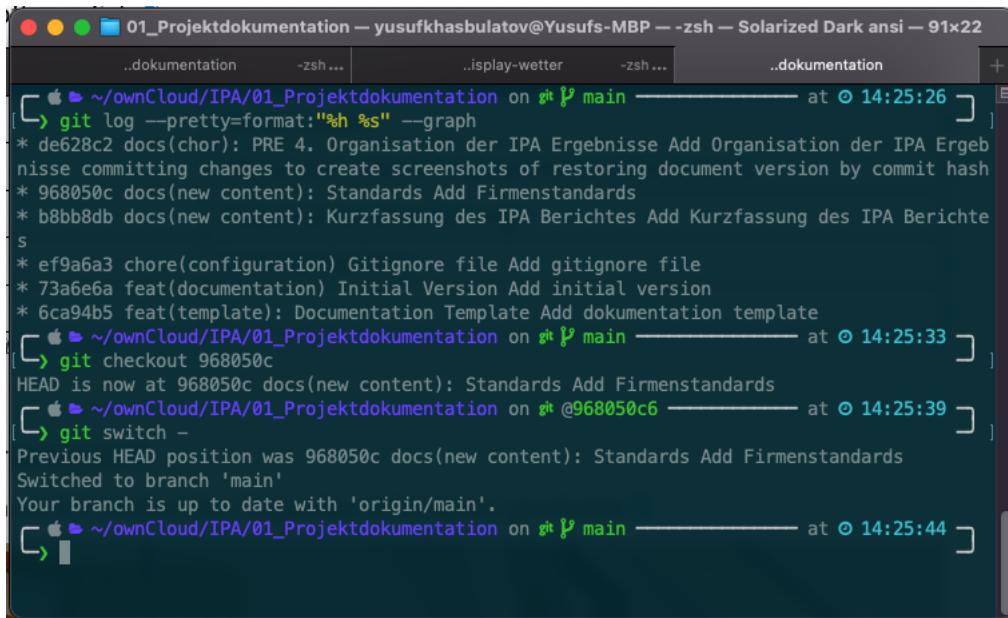
89grad GmbH GitLab Server

Die Wiederherstellung der Projektdokumentation über GitLab geht über mehrere Schritte. Bei der Verwendung der git CLI zuerst schaue ich welches HASH hat die gewünschte Version und dafür verwende ich «git log»-Befehl mit etwas Formatierung für besseres Übersicht.

```
«%h»-abgekürzter Commit-Hash  
«%s»-Betreff
```

Tabelle 4 git-CLI Log Formatierung

Danach kopiere ich das HASH und mache «git checkout» auf das HASH. Somit habe ich dann die Version, die unter dieser HASH gespeichert wurde.



The screenshot shows a terminal window titled "01_Projektdokumentation" with the command "git log --pretty=format:"%h %s" --graph". The output lists several commits:

- * de628c2 docs(chor): PRE 4. Organisation der IPA Ergebnisse Add Organisation der IPA Ergebnisse committing changes to create screenshots of restoring document version by commit hash
- * 968050c docs(new content): Standards Add Firmenstandards
- * b8bb8db docs(new content): Kurzfassung des IPA Berichtes Add Kurzfassung des IPA Berichte s
- * ef9a6a3 chore(configuration) Gitignore file Add gitignore file
- * 73a6e6a feat(documentation) Initial Version Add initial version
- * 6ca94b5 feat(template): Documentation Template Add dokumentation template

Afterwards, the user runs "git checkout 968050c", "git switch -", and "git switch main". The terminal window has tabs for ".dokumentation", "-zsh ...", ".isplay-wetter", "-zsh ...", and ".dokumentation". The background is Solarized Dark ansi.

Abbildung 12 Git CLI Wiederherstellung nach einem Tag

5. Projektvorgehen

Dieses Projekt wird nach der HERMES-Gibb-Projektmethode durchgeführt. Die HERMES-gibb Projektmethode wurde im Modul 306 kennengelernt. Es geht um eine «abgespeckte» Variante von HERMES 5.1. Folgende Punkte wurden im Modul 306 als Kernpunkte des HERMES-gibb genannt:

- Basiert auf Szenario «IT-Individualentwicklung»
- Nur ein Ergebnis pro Phase
- Projektantrag und -auftrag in der Studie integriert
- Optimiert auf Kleinprojekte wie M306 und IPA
- Fokus auf ein durchgängiges methodisches Vorgehen
- Kommentierte Vorlagen für alle Ergebnisse

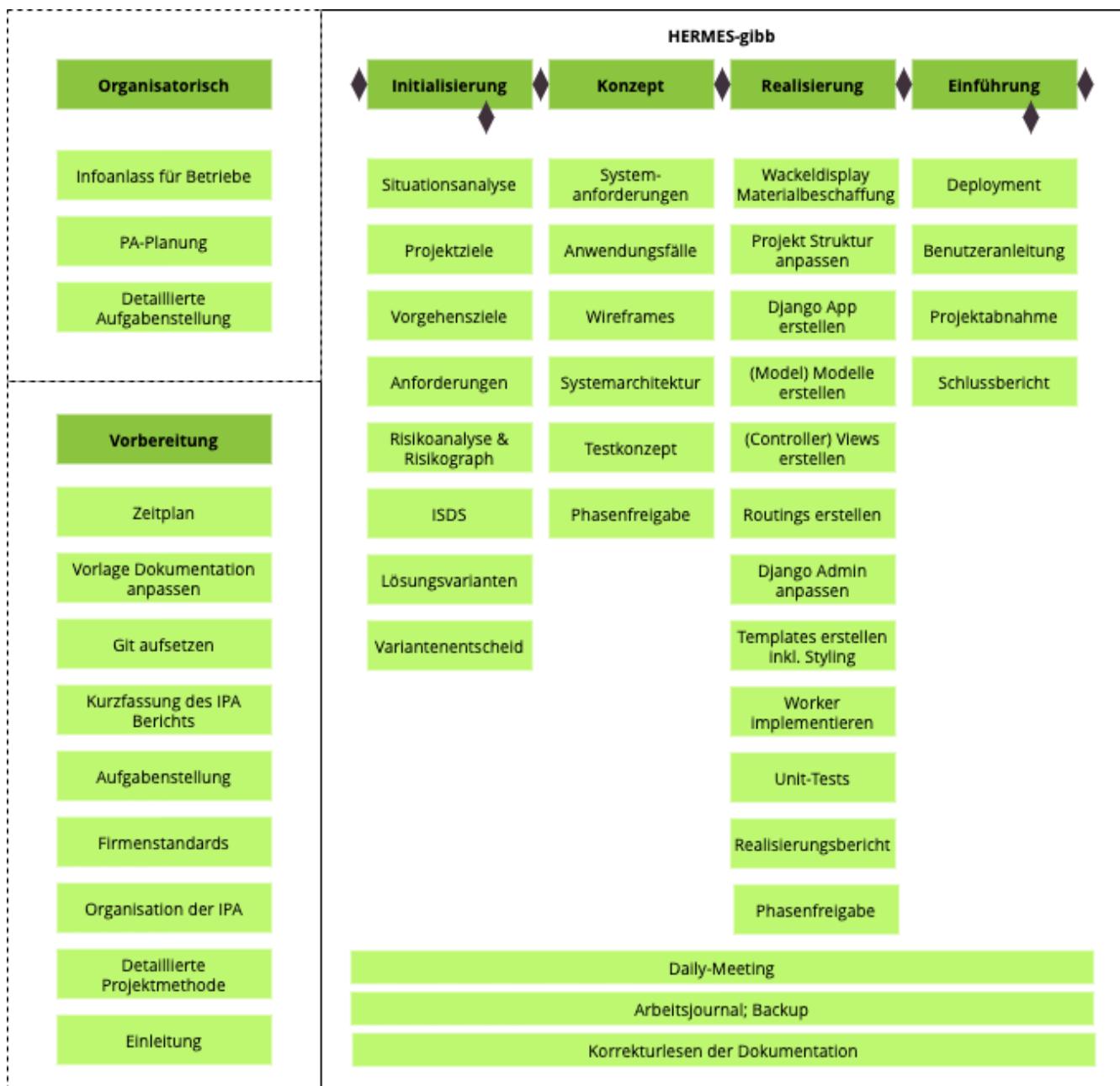


Abbildung 13 Gesamtübersicht des Projektplans

5.1. Projektvorgehensmodell

In dieser Arbeit werden sieben von HERMES empfohlene Meilensteine verwendet und zusätzlich definierte drei Meilensteine. In der nachstehenden Grafik sind die von HERMES empfohlenen Meilensteine dargestellt.

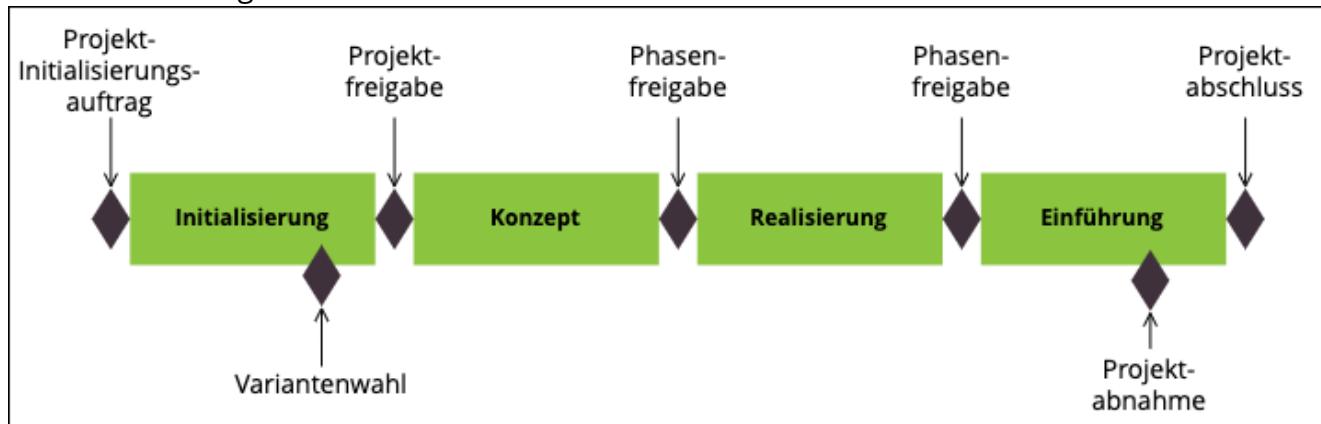


Abbildung 14 HERMES-gibb Projektvorgehensmodell

5.2. Szenario

HERMES-gibb wurde auf der Grundlage von HERMES 5.1 IT Individualentwicklung erstellt, und da es in diesem Projekt um die Entwicklung einer neuen Anwendung geht, wird dieses Szenario auch weiterhin verwenden.

Szenario	Beschreibung
IT-Individualentwicklung	HERMES-gibb basiert auf dem Szenario IT-Individualentwicklung.

Tabelle 5 HERMES-gibb Szenario

5.3. Phasen

Die vier von HERMES in diesem Projekt definierten Phasen werden im Folgenden beschrieben, es gab jedoch zwei weitere Phasen, die im [Projektablaufmodell](#) dargestellt sind. Bei diesen Phasen handelt es sich um Vorbereitungen, die nicht in die Initialisierungsphase einbezogen werden konnten, da sie nicht direkt zur Initialisierungsphase gehören.

Phase	Beschreibung
Initialisierung	Die Initialisierung schafft eine definierte Ausgangssituation für das Projekt und stellt sicher, dass die Projektziele mit den Zielen und Strategien der Organisation übereinstimmen. Die Projektgrundlagen und der Projektauftrag werden ausgearbeitet und die Entscheidung zur Freigabe des Projekts wird getroffen
Konzept	Die in der Initialisierungsphase ausgewählte Variante wird konkretisiert. Die Ergebnisse werden so detailliert ausgearbeitet, dass die Projektbeteiligten das Produkt oder IT-System auf einer verlässlichen Basis planen, anbieten und umsetzen können.
Realisierung	Das Produkt oder IT-System wird eingeführt und getestet. Die notwendigen Vorarbeiten werden geleistet, um die Einführungsrisiken zu minimieren.

Einführung	Der sichere Übergang vom alten zum neuen Zustand ist gewährleistet. Der Betrieb wird aufgenommen und durch das Projekt unterstützt, bis er stabil ist.
------------	--

Tabelle 6 HERMES-gibb Phasen

5.4. Module

Folgend sind die Module, die in diesem Projekt aufgeführt werden. Die Module sind vom HERMES-gibb für die Szenario IT-Individualentwicklung empfohlen.

Modul	Beschreibung
Projektsteuerung	<ul style="list-style-type: none"> Das Projekt initialisieren, führen und Vorgaben der Auftraggeber in Übereinstimmung halten. Risiken managen und Entscheide treffen. Das Projekt abschliessen.
Projektführung	<ul style="list-style-type: none"> Das Projekt planen, führen und in den definierten Rahmenbedingungen von Zeit zum Ziel bringen. Die Interessen der Auftraggeber kennen, die Kommunikation führen und Entscheide sicherstellen. Risiken managen, Probleme bewältigen und Erfahrungen berücksichtigen. Leistungen vereinbaren und steuern, die Qualitätssicherung führen.
Projektgrundlagen	<ul style="list-style-type: none"> Die Studie erarbeiten, damit der Variantenentscheid gefällt werden kann. Der Schutzbedarf analysiert. Die Voraussetzungen schaffen, um den Projektmanagementplan und den Projektauftrag zu erarbeiten.
Einführungsorganisation	<ul style="list-style-type: none"> Organisatorische Aufgaben und Massnahmen durchführen. Enthält die Abnahme.
IT-System	<ul style="list-style-type: none"> Das IT-System realisieren bzw. integrieren und dokumentieren. Die Systemanforderungen verfeinern.
IT-Betrieb	<ul style="list-style-type: none"> Das IT-System integrieren und aktivieren.
IT-Migration	<ul style="list-style-type: none"> Das Altsystem ausser Betrieb setzen.
Testen	<ul style="list-style-type: none"> Das Testen konzipieren, vorbereiten, durchführen und dokumentieren.
Informationssicherheit und Datenschutz	<ul style="list-style-type: none"> Anforderungen der Sicherheit und des Datenschutzes ermitteln, Risiken bewerten und Massnahmen zur Erfüllung der Anforderungen konzipieren und umsetzen. Das ISDS-Konzept erstellen und die Ergebnisse laufend dokumentieren.

Beschaffung	<ul style="list-style-type: none"> Die Beschaffung findet in der Phase Konzept statt. Wenn nötig können Beschaffungen auch in anderen Phasen durchgeführt werden.
-------------	--

Tabelle 7 HERMES-gibb Module

5.5. Meilensteine

Nachfolgend die Beschreibung der in diesem Projekt vorgesehenen Meilensteine. Wie bereits erwähnt, gibt es sieben Meilensteine, die von HERMES übernommen wurden, und drei selbst entworfene Meilensteine.

Nr.	Meilensteine	Beschreibung
1	Projektinitialisierungsauftrag	Hier wurde die IPA vom Valid Experten validiert, anders formuliert und dem Projektantrag wurde stattgegeben.
2	Vorbereitungen abgeschlossen	Der Teil 1 wurde in der Dokumentation fertig dokumentiert.
3	Variantenwahl und Projektfreigabe	Die Initialisierungsphase in der Dokumentation wurde geschrieben. Die Variantenentscheidung wurde getroffen und der Projektantrag wurde genehmigt.
4	Phasenfreigabe (Realisierung)	Die Wireframes und das Datenbankkonzept werden aufeinander abgestimmt. Testkonzept wird überprüft und Realisierungsphase wird freigegeben.
5	MVC Implementieren	Das Django MVC (bzw. MTV) ist implementiert und funktioniert.
6	Worker implementiert	Der Worker für das Wackeldisplay ist implementiert und getestet.
7	Phasenfreigabe (Einführung)	Der Realisierungsbericht ist als Teil des Schlussberichts erstellt. Die Anwendung wird getestet und abgestimmt. Die Phase Einführung wird freigegeben.
8	Benutzeranleitung erstellt	Die Benutzerhandbücher wurden erstellt und sind mindestens einmal durch eine Testperson durchgeführt worden.
9	Projektabnahme	Das Projekt funktioniert und die Unit Tests wurden fehlerfrei durchgeführt. Projekt läuft fehlerfrei und kann verwendet werden.
10	Projektabchluss	Am Ende der Phase Einführung wird nach erfolgreicher Betriebsaufnahme und nach dem Entscheid zur Abnahme der Projektabchluss gemacht.

Tabelle 8 Projekt Meilensteine

6. IPA Projektorganisation inkl. Projektrollen

6.1. Organigramm

Nachstehend ist das Organigramm und die Rollen, die an diesem Projekt beteiligt sind, aufgeführt. Ich habe die Rollen aus der HERMES-Rollenliste übernommen.

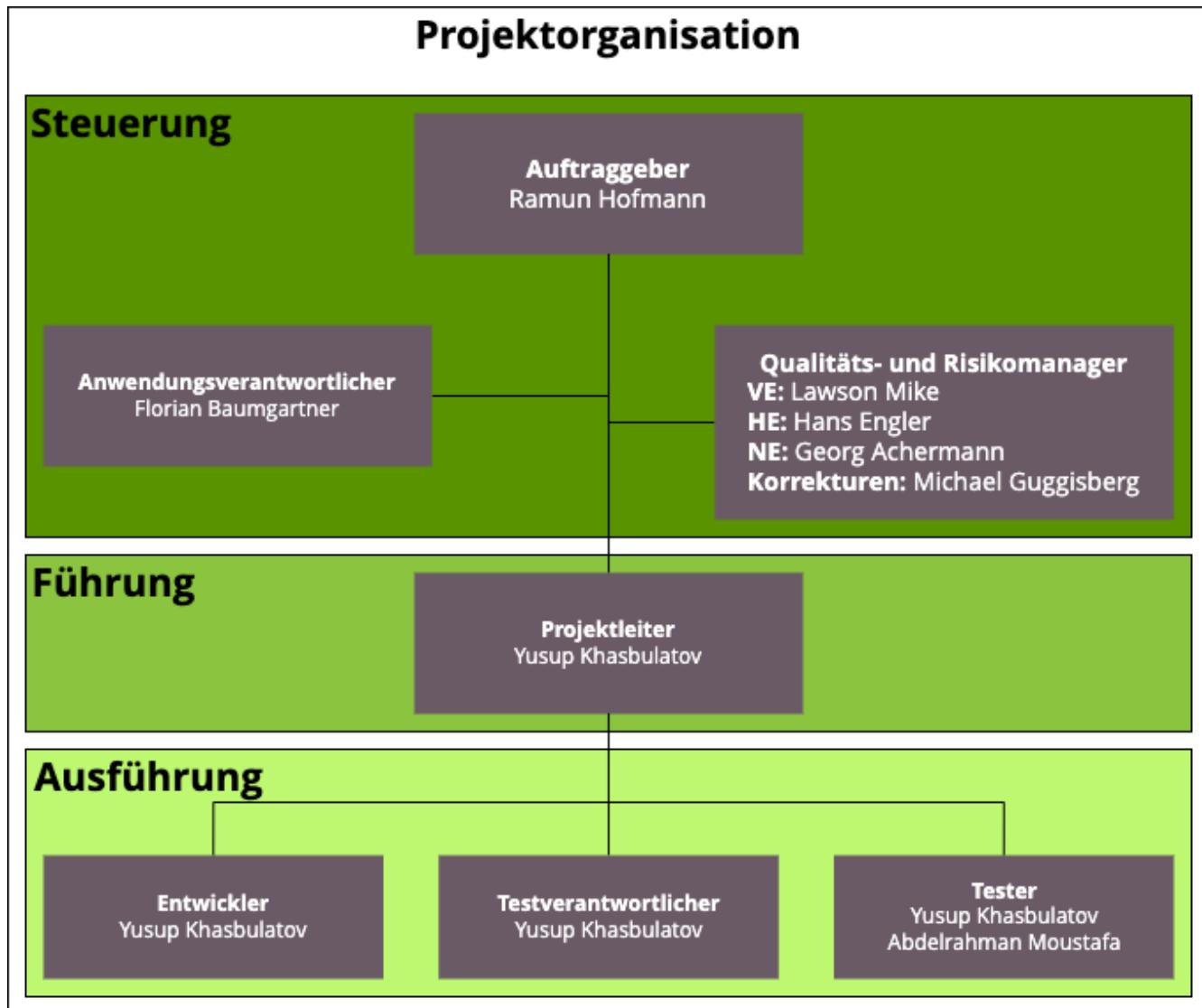


Abbildung 15 Projekt Organigramm

6.2. Projektrollen

Folgend sind die Beschreibungen von Projektrollen. Die Rollen wurden aus offiziellen HERMES 5.1 Dokumentation übernommen.

Person/Rolle	Beschreibung
Auftraggeber	Der Auftraggeber ist verantwortlich für die Ergebnisse des Projekts und die Erreichung der Ziele innerhalb des gesetzten Terminrahmens.
Anwendungsverantwortlicher	Der Anwendungsverantwortliche stellt den Unterhalt und die Weiterentwicklung sowie den sicheren und wirtschaftlichen

	Betrieb gemäss den entsprechenden Anforderungen und Vereinbarungen sicher.
Qualitäts- und Risikomanager	Der Qualitäts- und Risikomanager unterstützt den Auftraggeber mit einer unabhängigen Beurteilung des Projekts. Er gibt Empfehlungen für Massnahmen zur Erreichung der Projektziele ab.
Projektleiter	Der Projektleiter führt das Projekt im Auftrag des Auftraggebers. Er wird vom Auftraggeber ernannt und geführt.
Entwickler	Die Rolle Entwickler ist umfassend und bezeichnet den Produktentwickler und den IT-Entwickler. Der Entwickler entwirft, gestaltet und erstellt das Produkt bzw. das IT-System gemäss den Anforderungen unter der Führung des Projektleiters. Er integriert das Produkt bzw. das IT-System in die Umgebung des Betreibers.
Testverantwortlicher	Der Testverantwortliche konzipiert, plant und koordiniert das Testen. Er stellt sicher, dass die Testgrundlagen in Form des Testkonzepts erarbeitet werden, und überführt das Testen in den anschliessenden Betrieb.
Tester	Der Tester arbeitet bei der Erstellung der Testfallbeschreibungen mit, führt Tests durch und beurteilt und protokolliert die Ergebnisse.

Tabelle 9 Projektrollen

7. Zeitplan

Name: Yusup Khasbulatov		Datum: 02.05.2022	Version: 4	02.05.2022	05.05.2022	06.05.2022	09.05.2022	12.05.2022	13.05.2022	16.05.2022	19.05.2022	20.05.2022	23.05.2022	
	ID	Projektaufgabe	Total (h)	Differenz (h)	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist
Vorbereitung	1	Zeitplan	2	0	Soll 1	Ist 2								
	2	Dokumentation vorbereiten und Git aufsetzen	1	0	Soll 1	Ist 1								
	3	Kurzfassung des IPA Berichts	1.5	0.5	Soll 1	Ist 1	0.5							
	4	Aufgabenstellung und Firmenstandards	1	0	Soll 1	Ist 1								
	7	Organisation der IPA	1	0	Soll 1	Ist 0.5	0.5							
	8	Detaillierte Projektmethode	1.75	0	Soll 1.75	Ist 0.75	1							
	10	Einleitung	1.5	-1	Soll 0.5	Ist 0.5	1.5	2	0.5	0.5				
Initialisierung	11	Situationsanalyse und Projektziele	4.75	0.75	Soll 5.5	Ist 5.5	0.5	2	0.75	1.5				
	14	Anforderungen	1	0.5	Soll 1.5	Ist 1.5			0.5	0.5				
	15	Risikoanalyse & Risikograph ISDS	2	-1	Soll 1	Ist 1			1.5	0.5				
	17	Lösungsvarianten und deren Beschreibung	2	-1	Soll 1	Ist 1			1.5	0.5	3	1		
Konzept	19	Systemanforderungen	2	0.5	Soll 2.5	Ist 2.5			1	1				1.5
	20	Anwendungsfälle	1.5	1	Soll 2.5	Ist 2.5			1	1.5				
	21	Wireframes	2.5	-1.5	Soll 1	Ist 1			0.5	2				
	22	Systemarchitektur	2	0	Soll 2	Ist 2			0.5	1.5				
Realisierung	23	Testkonzept	3.5	0.5	Soll 1	Ist 1			1	0.5	0.5	1.5	2.4	
	24	Wackeldisplay Materialbeschaffung	1	0	Soll 1	Ist 1			1	1				
	28	MVC (bzw. MTV)	5.25	0.25	Soll 5.5	Ist 5.5			1	0.5	2	1.75	5	0.5
	30	Routing und Django Admin erstellen	0.5	0	Soll 0.5	Ist 0.5					0.5			
Einführung	33	Worker implementieren	4.5	-1	Soll 3.5	Ist 3.5			0.5	1.5	2	2	6	
	34	Unit-Tests. Korrekte Durchführung sicherstellen	2.5	2.5	Soll 5	Ist 5					1.5	1	2	0.5
	35	Realisierungsbericht	4.5	-3.5	Soll 1	Ist 1					1	2	1.5	7
	36	Deployment	2	-0.5	Soll 1.5	Ist 1.5					0.5	0.5	0.5	2
Phasenübergreifend	37	Benutzeranleitung	2	0	Soll 2	Ist 2					2	1	1	1
	38	Schlussbericht	6.5	-2	Soll 4.5	Ist 4.5					0.5	1	2	2
	39	Expertenbesuche; Daily-Meeting (Protokollieren)	4.75	3.25	Soll 8	Ist 8	0.25	1	0.25		0.5	0.5	0.5	0.25
	40	Korrekturlesen der Dokumentation	4.75	3.25	Soll 8	Ist 8	0.25	1.25	0.25	0.25	0.25	1	0.25	0.25
Meilensteine	41	Arbeitsjournal; Backup	9.25	-0.5	Soll 8.75	Ist 8.75	0.75	0.75	0.75	0.75	1	0.75	1	1.5
	42	Reserve	1.5	-0.5	Soll 1	Ist 1			0.25	0.25	1	0.5	1.5	0.75
	43	Abschluss IPA	1	0	Soll 1	Ist 1					1	1	1	10
	1	Projektinitialisierungsauftrag	Hier wurde die IPA vom Valid Experten validiert, anders formuliert und dem Projektantrag wurde stattgegeben.											
2		Vorbereitungen abgeschlossen	Der Teil 1 wurde in der Dokumentation fertig dokumentiert.											
3		Variantenwahl und Projekt freigabe	Die Initialisierungsphase in der Dokumentation wurde geschrieben. Die Variantenentscheidung wurde getroffen und der Projektantrag wurde genehmigt.											
4		Phasenfreigabe (Realisierung)	Die Wireframes und das Datenbankkonzept werden aufeinander abgestimmt. Testkonzept wird überprüft und Realisierungsphase wird freigegeben.											
5		MVC Implementieren	Das Django MVC (bzw. MTV) ist implementiert und funktioniert.											
6		Worker implementiert	Der Worker für das Wackeldisplay ist implementiert und getestet.											
7		Phasenfreigabe (Einführung)	Der Realisierungsbericht ist als Teil des Schlussberichts erstellt. Die Anwendung wird getestet und abgestimmt. Die Phase Einführung wird freigegeben.											
8		Benutzeranleitung erstellt	Die Benutzerhandbücher wurden erstellt und sind mindestens einmal durch eine Testperson durchgeführt worden.											
9		Projektabnahme	Das Projekt funktioniert und die Unit Tests wurden fehlerfrei durchgeführt. Projekt läuft fehlerfrei und kann verwendet werden.											
10		Projektabchluss	Am Ende der Phase Einführung wird nach erfolgreicher Betriebsaufnahme und nach dem Entscheid zur Abnahme der Projektabchluss gemacht.											

Total geplante Zeit	80.0	Legende
Total verfügbare Zeit	80.0	Soll Zeit
Total gebrauchte Zeit	80.5	Ist Zeit
		Meilenstein
		Expertenbesuch

8. Arbeitsjournale

Die Festlegungen dieses Dokuments gelten im Projekt.

Gemäss Art. 5 Absatz 2 der Wegleitung über die individuelle praktische Arbeit (IPA) an Lehrabschlussprüfungen des BBT vom 27. August 2001 gilt:

Die zu prüfende Person führt ein Arbeitsjournal. Sie dokumentiert darin täglich das Vorgehen, den Stand der Prüfungsarbeit, sämtliche fremde Hilfestellungen (auch das Internet ist eine Hilfestellung) und besondere Vorkommnisse wie z.B. Änderungen der Aufgabenstellung, Arbeitsunterbrüche, organisatorische Probleme, Abweichungen von der Soll-Planung.

Das Arbeitsjournal zur IPA ist zwingend zu führen und den Experten und Fachvorgesetzten vorzulegen. Das Arbeitsjournal ist täglich sinngemäss und korrekt auszufüllen.

Das Arbeitsjournal dient der Nachvollziehbarkeit der von den Lernenden ausgeführten Arbeiten und wird als Teil der IPA in die Bewertung mit einbezogen.

8.1. Tag 1: 02.05.2022

Tätigkeiten	Beteiligte Personen	Aufwand geplant (Std)	Aufwand effektiv (Std)
Zeitplan erstellen	Yusup Khasbulatov	2	2
Dokumentation vorbereiten und Git aufsetzen	Yusup Khasbulatov	1	1
Kurzfassung des IPA Berichts	Yusup Khasbulatov	1	1.5
Aufgabenstellung und Firmenstandards	Yusup Khasbulatov	1	1
Organisation der IPA	Yusup Khasbulatov	1	1
Detaillierte Projektmethode	Yusup Khasbulatov	0.75	0.75
Daily-Meeting (Protokoll führen)	Yusup Khasbulatov Ramun Hofmann	0.25	0.25
Arbeitsjournal; Backup	Yusup Khasbulatov	0.75	0.75
Korrekturlesen der Dokumentation	Yusup Khasbulatov Michael Guggisberg	0.25	0
Total:		8	8.25

Tagesablauf

Der Zeitplan war meine erste Aufgabe, und für seine Erstellung nahm ich als Beispiel den Zeitplan früherer Auszubildender der 89grad GmbH. Ich passte ihn auf mein Projekt an und begann, ihn auszufüllen. Es gelang mir, die geplante Zeit für die Erstellung des Zeitplans einzuhalten. Am Ende druckte ich ihn aus und klebte ihn an das Whiteboard. Dann begann ich, die Vorlage für die Projektdokumentation anzupassen. Ich habe ziemlich viel Zeit mit der Erstellung dieser Vorlage verbracht und hatte das Gefühl, dass ich sie sehr gut gemacht habe, sodass ich vor dem Hochladen der ersten Version nur noch Kleinigkeiten überprüfen musste, nämlich dass das Inhaltsverzeichnis korrekt und die Quelle für das Bild auf der Titelseite

vorhanden war. Jetzt war es an der Zeit, mit dem Schreiben zu beginnen, da alles dafür vorbereitet war. Für die Kurzfassung habe ich mehr Zeit gebracht als geplant, weil ich für Korrekturen gesorgt habe. Und das Nachdenken darüber, was genau zu beschreiben ist, hat mich auch Zeit gekostet. Die Beschreibung musste technisch nach dem Kriterium B1 erfolgen, was ich versucht habe zu berücksichtigen. Für die Organisation der IPA habe ich drei Stellen zum Speichern der Daten vorbereitet. Ich machte Fotos von meinem Arbeitsplatz und kopierte sie in die Dokumentation. Danach habe ich begonnen, die Projektmethode zu beschreiben, aber ich bin mit dieser Aufgabe noch nicht fertig, weil mir noch einige Informationen fehlen, wie man die Module in HERMES-gibb beschreibt. Ich werde dies morgen mit meinem Lehrer besprechen und die Modultabelle ausfüllen.

Für die Korrekturlesen haben wir entschieden, dass ich eine Liste mit allen zu korrigierenden Kapiteln erstelle und mit einer Kopie der Projektdokumentation mitschicke. Die Korrekturen werden in Form einer Review im MS Word gemacht. Daher kann ich selbst die Korrekturen nachher akzeptieren oder ablehnen.

Hilfestellung

- Ramun Hofmann half beim Ausdrucken des Zeitplans. Das Problem war, dass der Zeitplan nicht vollständig ausgedruckt wurde, weil er im A3-Format war.
- <https://www.deepl.com/translator>
Es ist meine Hilfestelle für die deutschen Korrekturen.
- <https://www.leo.org>
Es ist meine Hilfestelle für die deutschen Korrekturen, insbesondere um die korrekten Artikel nachzuschauen.
- <https://www.hermes.admin.ch/de/projektmanagement/verstehen/phasen-und-meilensteine/initialisierung.html>
Wie sieht die Beschreibung von Initialisierungsphase des offizielle HERMES 5.1 aus?
- <https://www.git-tower.com/learn/git/faq/git-checkout-commits>
Hier habe ich nachgeschaut, wie das «checkout» einer bestimmten GIT Commit gemacht werden muss.
- <https://stackoverflow.com/questions/36794501/disable-warning-about-detached-head>
Hier habe ich nachgeschaut, wie man die Nachricht von git CLI ausschaltet, die mich gestört haben.
- <https://www.tutorialspoint.com/how-to-tag-a-commit-in-git>
Hier habe ich geschaut, wie man einen Tag in git CLI erstellt. Dies habe ich schlussendlich jedoch doch nicht über das CLI gemacht, sondern über das GitLab GUI
- <https://stackoverflow.com/questions/51110685/how-to-split-terminal-into-panes-in-macbook>
Hier habe ich nachgeschaut, wie man das Terminal im MacOS Horizontal splittet und herausgefunden, dass es standartmässig nicht möglich ist. Daher habe ich weiter mit zwei Terminal-Tabs gearbeitet.
- https://doc.owncloud.com/server/next/admin_manual/configuration/files/file_versioning.html
Hier habe ich die Dokumentation von OwnCloud gelesen, um zu herauszufinden, wie man die Versionierung verwalten kann.
- <https://answers.microsoft.com/en-us/msoffice/forum/all/a3-printing-in-excel/ef3c13f3-dcd9-45cc-905c-ee22e28379d2>
In Excel auf dem MacOS gibt es kein vordefiniertes A3-Format zum Ausdrucken. Hier

habe ich herausgefunden, dass man ein eigenes Format definieren und die Grösse des A3 eingeben muss.

Reflexion

Positiv

Ich konnte die Aufgaben wie geplant alle realisieren. Nur einige Dinge gibt es, die ich noch abklären und danach nachführen muss.

Negativ

Der Ausdruck des Zeitplans auf A3 ging nicht.

Erkenntnisse

Wie man ein nicht bestehendes Format im Excel erstellen kann. Dies war hilfreich beim Ausdruck des Zeitplans.

Nächste Schritte

- Abklären mit dem Lehrer wie man Module im HERMES-gibb beschreiben soll.
- Detaillierte Projektmethode fertig machen.
- Technische Risikoanalyse erstellen. Diese werde ich jedoch zuerst mit dem Experten besprechen, da gemäss HERMES-gibb Initialisierungsphase auch eine Risikoanalyse erstellt werden muss.
- Erstellung der Einleitung gemäss den Kapiteln.
- Die Situationsanalyse erstellen.

Tabelle 10 Arbeitsjournal Tag 1

8.2. Tag 2: 05.05.2022

Tätigkeiten	Beteiligte Personen	Aufwand geplant (Std)	Aufwand effektiv (Std)
Expertenbesuch	Yusup Khasbulatov Ramun Hofmann Florian Baumgartner Hans Engler	1	1.5
Korrekturlesen der Dokumentation	Yusup Khasbulatov	0.25	1.25
Detaillierte Projektmethode	Yusup Khasbulatov	1	1
Einleitung	Yusup Khasbulatov	1.5	0.5
Situationsanalyse: IST-Zustand	Yusup Khasbulatov	3.25	2
Daily-Meeting (Protokollieren)	Yusup Khasbulatov Ramun Hofmann	0.25	0.75
Arbeitsjournal; Backup	Yusup Khasbulatov	0.75	1
Total:		8	8.75
Tagesablauf			
Heute habe ich Expertenbesuch gehabt. Wir haben über verschiedene Themen besprochen. Die genauen Details sind im Sitzungsprotokoll von 05. Mai 2022 aufgeführt. Es war nicht so einfach, mich darauf zu konzentrieren, was der Expert sagte und gleichzeitig Protokoll zu			

führen. Darum habe ich dieses Sitzungsprotokoll in einer der Notizen gemacht. Die Idee war, dass ich diese später ergänze. Jedoch war es sehr aufwendig, das Protokoll zu ergänzen. Der Hauptgrund war, dass der Expert erwähnt hat, dass es ein Beschlussprotokoll sein soll und nicht ein Verlaufsprotokoll. So musste ich das Protokoll nach dem Meeting mit dem Fachvorgesetzten umschreiben. Heute haben wir auch das monatliche Team-Meeting gehabt, welches ich nicht verpassen wollte. So hat sich mein Zeitplan mehr in den Nachmittag verschoben. Nach dem Team-Meeting habe ich die Einleitung geschrieben, obwohl ich noch die einige Teile zur Projektmethode fertig machen musste. Ich habe diese Aufgabe getauscht, da ich für die Einleitung viele Informationen aus dem 89grad GmbH nehmen konnte und da es schneller geht als die detaillierte Projektmethode zu schreiben. Ich werde mir noch eine Notiz schreiben, um dieses Kapitel mit Grafiken zu ergänzen.

Nach dem Austausch mit dem Lehrer war mir klar, dass die HERMES-gibb Projektmethode sehr flexibel ist und wo ich die Beschreibungen zu den Modulen finde. Diesen Teil habe ich heute ergänzt.

Die «Vorbereitungsphase»¹ wurde heute wie geplant abgeschlossen. Und ich habe angefangen die IST-Situation zu beschreiben.

Hilfestellung

- Ramun Hofmann – Antwort zu den Fragen, was ein Beschlussprotokoll ist und wie man ein solches schreibt.
- <https://www.hermes.admin.ch/de/projektmanagement/verstehen/module.html>
HERMES 5.1 Module. Ich habe nachgeschaut welche Module es gibt für welche Szenarien.
- <https://2022.pkorg.ch/dokument/503/view>
Dokumentationsvorgabe Pkorg. Ich habe die Beschreibung über die Kurzfassung des IPA-Berichtes nachgeschaut, um zu bestätigen, dass diese technisch sein soll.
- <https://www.deepl.com/translator>
Dieses Tool verwende ich täglich für die Korrekturen.
- <https://www.89grad.ch/softwareloesungen-kundenportale>
Ich habe die Beschreibung der 89grad GmbH durchgelesen und habe einige Stellen in diesem Bericht angepasst.
- <https://www.hermes-projektmanagement-training.ch>
verfügt über ein sehr gute HERMES 5 Zusammenfassung.
- <https://www.hermes.admin.ch/de/projektmanagement/verstehen/rollen.html>
Die HERMES Rollen habe ich nachgeschaut, um sie im Organigramm richtig darzustellen.
- <https://creately.com/blog/diagrams/network-diagram-guide-tutorial>
Hier habe ich die wesentlichen Netzwerkelemente in einem Diagramm nachgeschaut.

Reflexion

Positiv

Der Expertenbesuch ist gut gelaufen.

Der Expert war sehr zufrieden mit meinem Zeitplan.

Einen Meilenstein weniger, da ich alle Aufgaben für Meilenstein 2 erledigt habe.

Es gibt noch Verbesserungs-Möglichkeiten und -Ideen und ich habe mir Notizen dazu auf die Wandtafel geklebt.

Negativ

¹Ist keine HERMES-gibb Projektphase

Ich habe mir damit nicht gerechnet, dass der Termin nicht in geplante Zeit stattfindet und dies hat meinen Plan für heute ein wenig durcheinandergebracht.

Zudem habe ich viel Zeit für die Umschreibung des Sitzungsprotokolls verloren.

Erkenntnisse

Ich muss mich besser auf die Experten-Besuchstage vorbereiten und in der Lage sein meinen Plan zu ändern, wenn etwas in dem letzten Zeitpunkt sich ändert.

Nächste Schritte

- Situationsanalyse fertigmachen
- Anforderungen beschreiben
- Ziele beschreiben
- Risikoanalyse machen
- Variantenvergleich
- Projekt freigabe

Tabelle 11 Arbeitsjournal Tag 2

8.3. Tag 3: 06.05.2022

Tätigkeiten	Beteiligte Personen	Aufwand geplant (Std)	Aufwand effektiv (Std)
Situationsanalyse: - IST-Zustand - Stärken und Schwächen - Persönliche und Projektziele	Yusup Khasbulatov	1.5	3.5
Anforderungen	Yusup Khasbulatov	1	1.5
Risikoanalyse & Risikograph	Yusup Khasbulatov	2	1
Lösungsvarianten und deren Beschreibung	Yusup Khasbulatov	2	1
Variantenentscheidung und Projekt freigabe (Daily Meeting)	Yusup Khasbulatov Florian Baumgartner Ramun Hofmann	0.25	0.5
Korrekturlesen der Dokumentation	Yusup Khasbulatov Michael Guggisberg	0.25	0
Arbeitsjournal; Backup	Yusup Khasbulatov	0.75	1
Reserve	Yusup Khasbulatov	0.25	0
Total:		8	8.5

Tagesablauf

Da ich gestern nicht alle Teile der Situationsanalyse fertiggemacht hatte, habe ich heute mit diesem Teil angefangen. Es war mir wichtig, dass die Schwäche definiert sind, damit ich die Anforderungen korrekt definieren kann. Die Stärken habe ich jedoch auf einen späteren Zeitpunkt verschoben. Danach habe ich angefangen die Risikoanalyse zu machen. Wie gestern während dem Expertenbesuch besprochen habe ich die Kapitel «Technische Risikoanalyse» gelöscht und eine «Allgemeine Risikoanalyse» erstellt. Ich habe jedoch nicht möglichst viele

soziale Punkte erwähnt, sondern nur diejenigen, die ich wichtig finde. Den Risikograph (embedded Excel Tabelle) konnte ich nicht editieren, da MS-Word mir einen Fehler angezeigt. Ich habe diese Arbeit ebenfalls auf einen späteren Zeitpunkt verschoben. Es war mir wichtiger die Zeit zu nutzen, um eine Vergleichstabelle von verschiedenen API-Datenquellen zu erstellen. ISDS habe ich ebenfalls noch nicht beschrieben und werde dies nachholen. Auf der Suche nach der API-Datenquelle habe ich verschiedene Webseiten angeschaut, welche einen Vergleich zwischen verschiedenen Ressourcen machen. So konnte ich sehr schnell ganz viele Webseiten und deren API-Dokumentationen anschauen und mir eine Vorstellung machen welche Ressourcen ich für die Realisierung benötige. Dann habe ich meinen Fachvorgesetzten kontaktiert, um mit ihm die Kriterien für die Vergleichstabelle festzulegen. Der Vergleich ergab einen klaren Kandidaten und dieser wurde für das Projekt freigegeben.

Hilfestellung

- <https://www.tomorrow.io/blog/top-8-weather-apis-for-2022>
Eine TOP Liste von verschiedenen API-Quellen.
- <https://web.archive.org/web/20220407011409/https://rapidapi.com/blog/access-global-weather-data-with-these-weather-apis>
Eine TOP Liste von verschiedenen API-Quellen.
- <https://docs.djangoproject.com/en/4.0>
Django-Recherche zu den Themen Templates, Models und Meta-Klasse.
- <https://stackoverflow.com/questions/1483273/how-to-draw-a-classs-metaclass-in-uml>
Erklärt wie man die Meta-Klasse von Python in einen UML darstellen soll.
- <https://stackoverflow.com/questions/1301346/what-is-the-meaning-of-single-and-double-underscore-before-an-object-name>
Hier habe ich nachgeschaut, ob die Methoden in Python, die mit zwei Unterstrichen definiert sind, als Private Methoden dargestellt werden sollen.
- [https://de.wikipedia.org/wiki/Chen-Notation#Modifizierte_Chen-Notation_\(MC-Notation\)](https://de.wikipedia.org/wiki/Chen-Notation#Modifizierte_Chen-Notation_(MC-Notation))
Die Chen-Notation nochmals nachgeschaut.
- <https://www.conceptdraw.com/examples/how-to-show-unique-key-in-er-diagram>
Hier habe ich nachgeschaut, ob man irgendwie die «unique constraint» im UML darstellen kann.

Reflexion

Positiv

Projektfreigabe wurde gemacht und ich kann nun mit der Konzeptphase anfangen.

Negativ

Die Übernahme der Grafiken benötigte mehr Zeit als gedacht.

Erkenntnisse

Ich muss meinen Perfektionismus zurückhalten. Sonst verliere ich unnötig viel Zeit bei der Erstellung von Grafiken.

Nächste Schritte

- Konzeptphase anfangen
- Systemanforderungen und Benutzerschnittstellen

Tabelle 12 Arbeitsjournal Tag 3

8.4. Tag 4: 09.05.2022

Tätigkeiten	Beteiligte Personen	Aufwand geplant (Std)	Aufwand effektiv (Std)
Systemanforderungen	Yusup Khasbulatov	2	1
Anwendungsfälle	Yusup Khasbulatov	1.5	2.5
Wireframes	Yusup Khasbulatov	2.5	1
Systemarchitektur	Yusup Khasbulatov	1	0.5
Arbeitsjournal; Backups	Yusup Khasbulatov	0.75	1
Reserve	Yusup Khasbulatov	0.25	0
Daily-Meeting	Yusup Khasbulatov Ramun Hofmann	0	0.25
Total:		8	8.25
Tagesablauf			
Heute habe ich am Konzept des Projekts gearbeitet und konnte gute Fortschritte machen. Ich habe bereits viele Diagramme und Mockups mittels Draw.io erstellt und merke, dass ich mit diesem Tool mehr und mehr vertraut bin. Für die Black-Box Tests habe ich heute mit einem Mitarbeiter abgemacht, dass er das macht, und ich muss ihn nun als Tester ins Organigramm reinnehmen. Da ich das System im IST-Zustand sehr ausführlich beschrieben habe und sich nach meinen Überlegungen zum neuen System nicht viel ändert, habe ich weniger Zeit darauf verwendet, es erneut im Konzeptphase zu beschreiben.			
Hilfestellung			
<ul style="list-style-type: none"> • https://www.hermes.admin.ch/de/projektmanagement/verstehen/phasen-und-meilensteine.html Dokumentation über die Meilensteine durchgelesen, um die Beschreibung des Projektschlusses nachzulesen. • https://openweathermap.org/api OpenWeatherApi Recherche, um die Symbole für das Wetter nachzuschauen. • http://hilite.me Ein Tool für das Code Syntax Highlight. • https://de.wikipedia.org/wiki/White-Box-Test White-Box Test Beschreibung nachgeschaut. • https://de.parasoft.com/blog/how-to-write-test-cases-for-software-examples-tutorial Anleitung für die Beschreibung von Testfällen. 			
Reflexion			
<p>Positiv</p> <p>Ich habe einige Aufgaben schneller erledigt als geplant (was auch ein negativer Punkt ist, weil ich sie nicht richtig geplant habe). Hauptsächlich habe ich so geplant, weil ich dachte, dass ich mehr Zeit auf die Wireframes verwenden würde. Das war aber nicht der Fall, denn es gibt nur eine Schnittstelle, die ich sehr schnell definieren konnte. Ein weiterer Grund war, dass ich meinen Perfektionismus ausgeschaltet habe, weil ich in den letzten Tagen nicht zufrieden war, dass ich soviel Zeit auf Diagramme verwendete.</p>			

Die Zeit, welche ich heute gewonnen habe, habe ich für die Beschreibung des Testkonzepts verwendet. Morgen muss ich noch die Testfälle beschreiben, was aufwändig ist. Daher bin ich froh, dass ich bereits Vorarbeiten erledigen konnte.

Negativ

Mit meinem Zeitplan für heute bin ich nicht zufrieden, da nicht alle Termine korrekt aufgeführt waren. Das Daily Meeting war nicht im Zeitplan eingetragen, obwohl ich einen Kalender-Eintrag für Herrn Hofmann erstellt hatte. Ich sollte entweder ein Daily im Zeitplan einplanen oder keinen Kalendereintrag erstellen.

Erkenntnisse

Die Aufgabenstellung habe ich nicht richtig verstanden. Nach dem heutigen Daily-Meeting war mir klar, dass die Trend-Spalte nicht die Daten für die kommenden drei Tage zeigen soll, sondern den Durchschnitt für die nächsten drei Tage.

Nächste Schritte

- Die Testfälle weiter ergänzen
- In der Dokumentation Korrekturen machen
- Kurt für die Materialbeschaffung kontaktieren
- Die Systemarchitektur ergänzen

Tabelle 13 Arbeitsjournal Tag 4

8.5. Tag 5: 12.05.2022

Tätigkeiten	Beteiligte Personen	Aufwand geplant (Std)	Aufwand effektiv (Std)
Systemarchitektur	Yusup Khasbulatov	1	1.5
Testkonzept	Yusup Khasbulatov	3	2
Korrekturlesen	Yusup Khasbulatov	0.25	1
Für Phasenfreigabe treffen (Daily Meeting)	Yusup Khasbulatov Florian Baumgartner Ramun Hofmann	0.25	1.5
Materialbeschaffung (Symbole erstellen, Kurt kontaktieren)	Yusup Khasbulatov	1	1
MVC (Django Applikation)	Yusup Khasbulatov	1.5	0
Arbeitsjournal; Backups	Yusup Khasbulatov	1	1
Total:		8	8

Tagesablauf

Heute habe ich die Korrekturen, die Herr Guggisberg gemacht hat, durchgeschaut und angepasst. Es war jedoch aufwendiger als ich dachte. Grund dafür war, dass einige Formatierungen nicht richtig dargestellt wurden, und so konnte ich am Anfang den Unterschied nicht direkt erkennen. Ich habe die «Formatting Marks» aktiviert und das Problem gesucht, aber es war einfach nicht klar, weshalb die Formatierung nicht korrekt dargestellt wurde. Dann habe ich bemerkt, dass einige Abbildungsbeschreibungen nicht richtig formatiert und die Paragraf Margins zu hoch eingestellt waren. Sobald ich beide Fehler behoben hatte,

war das Problem danach gelöst. Anschliessend habe ich wieder eine Version im GitLab erstellt und angefangen die Systemanforderungen zu ergänzen. Der Verweis auf den IST-Zustand war nicht sinnvoll in der Situationsanalyse, ohne den IST-Zustand weiter zu ergänzen. Darum habe ich zuerst dieses Kapitel noch genauer beschrieben und dann die Systemarchitektur in der Konzeptphase weiterbearbeitet.

Für das Testkonzept habe ich weitere Testfälle beschrieben und werde es später noch weiter ausarbeiten. Dafür habe ich mir eine Notiz erstellt.

Die Phasenfreigabe lief ganz anders ab als erwartet. Ich habe dafür halbe Stunde geplant und dachte, dass meine Konzepte akzeptiert werden und ich weiter Arbeiten werde, aber dies war nicht der Fall. Die Mockups und Symbolen Konzepte wurde akzeptiert und wir haben aber beschlossen, dass ich noch Datenbank Konzept erstellen soll für die Abspeicherung der Daten. Die Idee dahinter war, dass das Wackeldisplay die zuletzt gespeicherten Daten anzeigt, sollte die OpenWeatherAPI nicht verfügbar sein. Zudem legten wir während dieser Sitzung das Datenformat fest. Genauer habe ich es im [Sitzungsprotokoll 6](#) beschrieben.

Hilfestellung

- <https://officebeginner.com/msword/how-to-print-a-word-document-without-comments/#:~:text=Click%20on%20the%20Settings%20%2D%3E%20Print,comments%20from%20the%20print%20view.>
Hier habe ich nachgeschaut, wie man Dokumente ohne Kommentare in ein PDF exportieren kann.
- <https://support.microsoft.com/en-us/office/print-a-document-in-word-for-mac-2d92b498-01a2-4a88-b833-83027173ae9c>
Hier habe ich nachgesehen, wie man die "Querformatausrichtung" ausdrucken kann. Leider hat es bei mir nicht funktioniert. Ich bin es gewohnt, MS-Office auf Windows zu verwenden. Aber auf MAC OS sind einige Dinge schwierig zu erledigen.
- <https://www.srf.ch/meteo/wetter/Bern/46.9471,7.4441?geolocationNameId=b37f9339c3e89d6ef433634ef14933b6>
<https://weather.com/weather/monthly/l/d171b86d9013f975a1f32bd871beadeb741230ee1db7d4a182a9ea1a641d9e56>
<https://www.meteoschweiz.admin.ch/home/klima/klima-der-schweiz/jahresverlauf-temperatur-sonne-niederschlag.html>
<https://www.wetter.com/reise/klima/klimatabelle/schweiz-bern-CH0CH0324.html>
Unter diesen Links habe ich recherchiert, wie die durchschnittliche Niederschlagsmenge in der Schweiz ist, um festzulegen, welche Intervalle bei den Niederschlagswerten auf dem Wackeldisplay angemessen sind. Schliesslich habe ich auf der SRF-Webseite gesehen, dass die Werte ungefähr zwischen 0-10mm/h liegen können.
- <https://git-scm.com/docs/pretty-formats>
Hier habe ich die Dokumentation dargelesen, um zu herauszufinden wofür genau «%h» und «%s» im git log Kommando verwendet werden.

Reflexion

Positiv

Ich konnte heute einige Kapitel ergänzen. Somit habe ich nun weniger Post-its auf der Wand.

Negativ

Ich muss nun Überlegungen zu einer geeigneten Datenbank für die Wetterdaten anstellen. Diesen zeitlichen Aufwand hatte ich bis jetzt nicht geplant. Andererseits hoffe ich, dass der Umstand, dass ich flexibel auf Ungeplantes reagieren kann, durch die Experten gut bewertet wird.

Erkenntnisse

In Adobe Acrobat kann man die Seiten einrichten. Das ist hilfreich, wenn man die Dokumentation ins PDF-Format exportieren möchte. Vor allem auf dem MAC OS funktioniert der Export bei unterschiedlicher Layout-Ausrichtung nicht richtig.

Nächste Schritte

- Django Applikation erstellen
- Projekt Struktur anpassen
- bootstrap scss Dateien herunterladen
- Verschiedene Tools für besseres Code Styling installieren:
 - djLint
 - black
 - flake8
 - django-sass-processor
- Schauen wie git pre-commit funktioniert, damit die benötigten Befehle immer automatisch ausgeführt werden

Tabelle 14 Arbeitsjournal Tag 5

8.6. Tag 6: 13.05.2022

Tätigkeiten	Beteiligte Personen	Aufwand geplant (Std)	Aufwand effektiv (Std)
MVC (bzw. MTV)	Yusup Khasbulatov	3.75	3
Datenbank Konzept (neue Tabellen)	Yusup Khasbulatov	0	1.5
Routing und Django Admin	Yusup Khasbulatov	0.5	0.5
Worker Implementieren	Yusup Khasbulatov	0.5	0
Zweiter Expertenbesuch	Yusup Khasbulatov Ramun Hofmann Hans Engler Georg Achermann	1	1
Daily Meeting	Yusup Khasbulatov Ramun Hofmann	0.25	0.5
Korrekturen der Dokumentation	Yusup Khasbulatov	1.25	1
Arbeitsjournal; Backup	Yusup Khasbulatov	0.75	0.5
Total:		8	8
Tagesablauf			
Heute habe ich die Models implementiert und den Django-Admin so angepasst, dass ich jetzt die Daten eingeben und ändern kann. Die Templates (Views) und Views (Controller) ² konnte ich leider noch nicht implementieren, da ich für die Implementierung der Models mehr Zeit gebraucht habe als ich dachte. Bei der täglichen Besprechung wurde mir klar, dass das Datenbank Konzept, das ich im Kopf hatte, nicht 100%ig richtig war. Ich benötige eine			

² Im Django Welt sind Templates als Views und Views als Controllers zu betrachten.

zusätzliches Zwischentabelle in der Datenbank für die Wetter-Symbole, weil ich sonst redundante Daten aufbaue. Deshalb habe ich nach der Besprechung ein Diagramm erstellt, das mir als Hilfsmittel bei der Modellerstellung dienen wird. Das Routing habe ich derzeit als Platzhalter definiert und muss später implementiert werden.

Während des Expertenbesuches wurden mir Fragen zur Projektstruktur, zu den Testmethoden, zur Projektdurchführungsmethode und zu den Datenbankgrundlagen gestellt, die ich beantworten konnte. Wir sprachen über den bevorstehenden Präsentationstag und wie es weitergeht. Die Experten wollten wissen, wo ich Schwierigkeiten habe und wie ich sie löse. Weitere Beschlüsse habe ich im [Sitzungsprotokoll 8](#) festgehalten.

Hilfestellung

- <https://en.wikipedia.org/wiki/Model%20view%20controller>
Hier habe ich das WIKI für MVC-Muster durchgelesen, um die Theorie aufzufrischen.
- <https://stackoverflow.com/questions/3855127/find-and-kill-process-locking-port-3000-on-mac>
Hier habe ich nachgeschaut, wie man einen besetzten Port freischaltet, um damit den zugehörigen Prozess zu stoppen.
- <https://www.epochconverter.com>
Dieses Tool habe ich verwendet, um den Zeitstempel in ein Datum umzuwandeln.
- <https://www.cyberciti.biz/faq/how-to-check-os-version-in-linux-command-line>
Hier war ich auf der Suche, wie man die Version der Linux-Distribution anzeigen kann. Ich brauche es, um das GitLab CI anzupassen und genau das gleiche Betriebssystem in einer virtuellen Maschine zu verwenden. Das System läuft auf der Debian-Version 11.
- <https://stackoverflow.com/questions/3098681/is-there-a-naming-convention-for-django-apps>
Ich wollte die bestehende «Columns»-Applikation umbenennen und den Grossbuchstaben durch einen Kleinbuchstaben ersetzen und habe nach einer offiziellen Namenskonvention für Django-Projekte gesucht. Der Grund dafür war, dass ich nach der Umbenennung viel Zeit bei der Fehlersuche verloren hatte. Lokal funktionierte das Projekt einwandfrei. Aber wenn ich die GitLab-Pipeline laufen liess, konnte GitLab keine neuen Module mit klein geschriebenen «columns» finden. Ich habe jedoch die Ursache des Problems nicht gefunden. Weil ich nicht noch mehr Zeit darauf verwenden wollte, habe ich die Änderungen rückgängig gemacht. Wenn ich am Schluss des Projekts noch Zeit habe, werde ich erneut versuchen, diese Anwendung umzubenennen.
- <https://pre-commit.com>
Ich hatte das Projekt so konfiguriert, dass jedes Mal, wenn ich die Änderungen übertrage, automatisch Anpassungen am Kode-Stil vorgenommen werden. Nachdem ich sie gepusht hatte, stieß ich auf das obige Problem und machte die Änderungen wie geschrieben wieder rückgängig.

Reflexion

Positiv

Ich habe nach dem Expertenbesuch ein gutes Feedback von meinen Vorgesetzten erhalten, was mich sehr gefreut hat. Vor allem, weil ich heute den ganzen Tag sehr nervös und gestresst war.

Negativ

Die Tatsache, dass ich die Anwendung nicht umbenennen konnte, hat mich sehr erstaunt. Ich mag es nicht, dass sie grossgeschrieben ist, da alle anderen Dateien klein geschrieben sind. Da

die Umbenennung weitere Fehler verursachte, habe ich die Arbeit am aktuellen Repository komplett aufgegeben und einen neuen Fork erstellt, ohne all diese Code-Formatierungsprogramme zu installieren oder die Anwendung umzubenennen. Es hat mich gestresst, Zeit mit der Lösung eines Problems zu verbringen, das am Ende nicht funktioniert hat, und ich wollte nicht riskieren, noch mehr Zeit damit zu verschwenden. Letztendlich kann ich die Codeformatierung manuell aufrufen, um zu prüfen, ob die Kodierungskonventionen erfüllt sind.

Erkenntnisse

- Während des Expertenbesuchs baten mich die Experten, die Schriftarten in der PyCharm-IDE zu vergrößern, was ich nicht konnte, weil ich es zuvor noch nie gemacht hatte. Dank meines Vorgesetzten weiß ich jetzt, dass die Funktion «Pitch» dafür geeignet ist.
- Es ist besser, während der IPA nicht zu versuchen, früher nicht verwendete Tools wie pre-commit zu verwenden.

Nächste Schritte

- Templates implementieren
- Workers für Daily Request und Hourly Request implementieren
- Unit-Tests schreiben
- Worker für das Wackeldisplay implementieren

Tabelle 15 Arbeitsjournal Tag 6

8.7. Tag 7: 16.05.2022

Tätigkeiten	Beteiligte Personen	Aufwand geplant (Std)	Aufwand effektiv (Std)
Worker implementieren	Yusup Khasbulatov	4	3.5
Unit-Tests schreiben und durchführen, allfällige Fehler im Code korrigieren	Yusup Khasbulatov	1.5	0
MVC – Komponenten (Templates «View»)	Yusup Khasbulatov	0	1
Daily-Meeting	Yusup Khasbulatov Ramun Hofmann	0.5	0.5
Korrekturlesen der Dokumentation	Yusup Khasbulatov	1	0
Arbeitsjournal und Backup	Yusup Khasbulatov	1	1.5
Systemanforderungen: Datenkonzept und Recherche für die Niederschlagswerte.	Yusup Khasbulatov	0	1.5
Total:		8	8

Tagesablauf

Ich habe heute einen sehr produktiven Tag gehabt, so dass ich nicht bemerkt habe, wie die Zeit verging. Ich konnte heute alle Worker implementieren. Insgesamt habe ich 3 Worker und 2 Skripte geschrieben, die die Erstellung der initialen Daten automatisieren. Zudem habe ich noch eine Recherche zu den Niederschlagswerten in den letzten 12 Monaten gemacht. Dieser Teil war spannend, da ich die Werte indirekt abfragen musste, weil es mit einer direkten Abfrage nicht funktioniert hatte. Heute habe ich meine Dokumentation lediglich für die

Erfassung des Arbeitsjournals geöffnet. Alles, was ich heute gemacht habe, werde ich am Donnerstag dokumentieren. Die Templates konnte ich heute implementieren, obwohl es nicht geplant war.

Hilfestellung

- <https://www.youtube.com/watch?v=K2PiHznrPBs>
Hier habe ich geschaut, wie «Broken Access Control» funktioniert. Ich war heute in der Lage, Daten aus einer API-Quelle zu lesen. Dafür musste ich aber eigene Header in der Anfrage erstellen und ich wollte verstehen, ob dies als «Broken Access Control» zählt. Grundsätzlich gilt es, wenn MeteoSchweiz standardmäßig keinen Zugriff gewähren will, aber wenn man einen Header setzt, wird der Zugriff möglich. Meiner Meinung nach sollte MeteoSchweiz einen Reverse Proxy verwenden, um zu sehen, woher die Anfragen wirklich kommen. Aber es kann natürlich sein, dass sie diesen Mechanismus mit Referenz-Header als «Schutz» nicht verwenden, sondern sie haben andere Gründe dafür.
- <https://chrome.google.com/webstore/detail/modheader/idgpnmonknjnojddfkpgkijpfnnfcklj?hl=en>
Mit diesem Tool konnte ich eigene Header erstellen und die «Access Denied» Fehlermeldung von der Wetterdaten-Webseite umgehen.
- <https://www.meteoschweiz.admin.ch/home/messwerte.html?param=messwerte-niederschlag-1d&station=BER&chart=day>
Hier habe ich die Niederschlagswerte nachgeschaut.
- <https://dev.to/vigo/django-model-best-practices-3e8e>
Hier habe ich geschaut, ob es «Best practices» gibt, um Kommentare zu machen.
- <https://docs.djangoproject.com/en/4.0/ref/models/querysets/#get-or-create>
Hier habe ich mich in der Django Dokumentation über die get_or_create Methode informiert.
- <https://stackoverflow.com/questions/70185513/runtimewarning-datetimefield-model-date-received-a-naive-datetime-while-time-zo>
Das System hat mir eine Datum-Warnung ausgegeben. Hier habe ich eine Lösung gefunden und gesehen, dass Django über eine Methode verfügt, um Datum-Objekte zu verarbeiten.
- <https://mdbootstrap.com/docs/standard/tools/builders/table>
Dieser Tool habe ich verwendet, um Django-Tabellen zu generieren. Es ist so viel einfacher die Tabelle zu erstellen und dann im Code anzupassen.
- <https://getbootstrap.com/docs/5.2/components/modal/#toggle>
Ich musste heute eine «Modal» Komponente erstellen. Da ich Bootstrap als CSS-Framework verwende, habe ich hier die Dokumentation zur «Modal» Komponente nachgelesen.

Reflexion

Positiv

Ich habe alle Worker implementiert und muss nun den Code noch schöner machen (mehr kommentieren, nach Styling-Fehler suchen, usw.) Die Worker muss ich noch mit dem Gerät testen.

Negativ

Ich habe heute sehr wenig Pause gemacht und merke es sehr. Dafür habe ich weniger zu erledigen im Projekt.

Erkenntnisse

Das Tool für eigene Headers im Request habe ich zuvor nie verwendet. Es ist ein sehr nützliches Tool. Ich hätte keine Skripte schreiben müssen, hätte ich das Tool bereits früher gekannt.

Und unbedingt mehr Pause machen.

Nächste Schritte

- Unit Tests schreiben
- Workers testen

Tabelle 16 Arbeitsjournal Tag 7

8.8. Tag 8: 19.05.2022

Tätigkeiten	Beteiligte Personen	Aufwand geplant (Std)	Aufwand effektiv (Std)
Unit Tests schreiben und durchführen	Yusup Khasbulatov	1	4.5
Realisierungsbericht	Yusup Khasbulatov	4.5	0
Phasenfreigabe (Einführung) (Daily Meeting)	Yusup Khasbulatov Ramun Hofmann	0.5	0.5
Korrekturlesen	Yusup Khasbulatov	1	1
Arbeitsjournal	Yusup Khasbulatov	1	1
Deployment	Yusup Khasbulatov	0	1
Total:		8	8

Tagesablauf

Heute habe ich viele Unit-Tests geschrieben. Mein Ziel war es, mehr als 90% Abdeckung zu erreichen. Allerdings habe ich die Unit-Tests nur für das Wetter-Paket geschrieben, und es reicht nicht aus, um weitere Tests für das Columns-Paket zu schreiben. Ich denke, damit kann ich leben, denn das Columns-Paket hat bisher auch ohne Tests gut funktioniert, und ich muss anderen Dingen den Vorrang geben.

Zur besseren Übersicht habe ich die Bibliothek «Coverage» installiert und die Django manage.py Skript so angepasst, dass sie bei jeder Ausführung eines Testbefehls automatisch einen Coverage Report erstellt. Außerdem habe ich in meiner lokalen Umgebung den Coverage HTML Generator verwendet, der die Erstellung von Unit-Tests sehr einfach macht, indem er anzeigt, welche Zeile während der Testausführung nicht getestet wurde.

Nachdem alle Tests erfolgreich waren, habe ich meinen Locales Branch, in dem ich die Tests implementiert habe, mit dem Develop Branch zusammengeführt. In GitLab CI funktionierten beim ersten Versuch alle Tests nicht. Der Ordner "Scripts" befand sich in meiner globalen gitignore-Datei und ich habe nicht bemerkt, dass er nicht versioniert war. Deshalb habe ich mich entschlossen, den Scripts-Ordner in utils umzubenennen.

Da ich heute viele Unit-Tests schreiben könnte habe ich das Projekt noch deployed, nach dem Deployment habe ich noch ein paar Fehlermeldungen gesehen die Werde ich morgen genauer anschauen und meine Tests verbessern damit solche Fälle auch abgedeckt werden.

Hilfestellung

- <https://stackoverflow.com/questions/64072833/does-django-testing-have-a-breakdown-similar-to-setup-that-is-run-after-all>
Hier habe ich gesucht, ob es eine Möglichkeit gibt, zusätzliche Dinge zu tun, nachdem alle Tests ausgeführt wurden. Mein Ziel war es, eine Datei zu löschen, die während der Ausführung der Tests erstellt werden könnte.
- <https://coverage.readthedocs.io/en/6.3.3/install.html>
Hier habe ich die Coverage-Dokumentation gelesen und geschaut, wie man die Konfiguration für die Coverage vornimmt. Es hat mir geholfen, die .coveragerc-Datei zu erstellen und einige Dinge für die Coverage zu ignorieren.
- <https://stackoverflow.com/questions/27064807/python-coverage-py-exclude-lines>
Hier habe ich geschaut, wie man bestimmte Code Zeile aus dem Coverage Rapport entfernen kann.
- <https://adamj.eu/tech/2019/04/30/getting-a-django-application-to-100-percent-coverage>
Hier war auch ein guter Guide für die Konfiguration für den Coverage. Mir hat besonders geholfen der Teil wo es um manage.py handelte.
- <https://stackoverflow.com/questions/11109468/how-do-i-import-the-django-doesnotexist-exception>
Hier habe ich nachgeschaut, wie man auf eine Exception testet. Es hat mir nicht viel geholfen, denn in meinem Code gibt es viele Exceptions, die ich behandle. An mehreren Stellen habe ich nach bestimmten Exceptions gesucht und wenn sie auftreten, schreibe ich verschiedene Logs. Ich musste also die Tests so erstellen, dass sie die Ausgabe der Logs überprüfen.
- <https://www.philipp-doblhofer.at/en/blog/automatic-changelog-and-versioning-with-git>
Da ich den Commit nach Conventional Commit Struktur gemacht habe, habe ich hier nachgeschaut, wie man nun einen automatischen Release Generator machen kann.

Reflexion

Positiv

Ich konnte heute sehr viele Unit-Tests schreiben (aber Unit-Tests sind nie zu viele)

Negativ

Leider blieb mir keine Zeit, den Realisierungsbericht zu schreiben, da ich mehr Zeit für die Unit-Tests aufgewendet habe, um eine Testabdeckung von mehr als 90% zu erreichen.

Erkenntnisse

Ich habe heute die Möglichkeit der Coverage entdeckt, um das HTML zu erstellen, das hat mir wirklich sehr geholfen. Sonst wäre es mühsam, nach Zeilen zu suchen, die im Rapport angezeigt werden.

Nächste Schritte

- Testprotokoll
- Black-Box Tests
- Projekt auf dem Server noch mal testen

Tabelle 17 Arbeitsjournal Tag 8

8.9. Tag 9: 20.05.2022

Tätigkeiten	Beteiligte Personen	Aufwand geplant (Std)	Aufwand effektiv (Std)
Black-Box Tests	Yusup Khasbulatov	0	0.5

Realisierungsbericht	Yusup Khasbulatov	0	1
Deployment	Yusup Khasbulatov	2	0.5
Benutzeranleitung	Yusup Khasbulatov	2	1
Schlussbericht	Yusup Khasbulatov	2.5	1.5
Daily-Meeting	Yusup Khasbulatov Ramun Hofmann	0.25	0.25
Korrekturlesen	Yusup Khasbulatov	0.25	1.25
Arbeitsjournal	Yusup Khasbulatov	1	0.5
Total:		8	8

Tagesablauf

Heute war mein vorletzter Tag und ich habe fast den ganzen Tag mit Schreiben verbracht. Ich konnte einige Teile des Benutzerhandbuchs fertigstellen und werde es am Montag abschliessen.

Die Black-Box-Tests sind endlich vorbei, und sie waren alle erfolgreich, mit einigen Bemerkungen, die Herr Moustafa in das Testprotokoll geschrieben hat. Vor den Blackbox-Tests musste ich noch einen Fehler im Backend beheben, den ich gemacht hatte. Aber das war keine grosse Sache und es ging nur um eine falsche Datenabfrage. Da ich gestern viele Unit-Tests geschrieben habe, geht das Deployment schneller, weil ich so sicherstellen kann, dass ich keine Fehler im Code habe.

Hilfestellung

- <https://www.qr-code-generator.com>
Ich habe diese Webseite für den QR-Code-Generator verwendet, aber ich werde den QR-Code anders machen, weil diese Webseite den QR-Code auf ihrem eigenen Proxy erstellt hat, der dann auf die gewünschte Webseite weiterleitet.
- <https://support.microsoft.com/en-us/office/create-a-table-ofAuthorities-ddd126ae-52bc-4299-9558-06dd0e4fe8c0>
Hier habe ich geschaut, wie man ein Abkürzungsverzeichnis im Word erstellen kann.

Reflexion

Positiv

Ich habe das gesamte Dokument heute durchgelesen und die Korrekturen vorgenommen.

Negativ

Mein Laptop war heute den ganzen Tag sehr langsam. Ich habe alle Dateien gespeichert und den Laptop neu gestartet, aber danach war er immer noch sehr langsam. Nachdem ich auf Speichern klicke, dauert es 10-20 Sekunden, bis das Dokument gespeichert ist.

Erkenntnisse

Ich habe die Funktion «Replace» in Word entdeckt. Ich benutze diese Funktion in PyCharm oder VS-Code sehr oft und wusste nicht, dass sie auch in Word möglich ist. So konnte ich z.B. die falsche Benennung des ESP-Mikrocontrollers korrigieren.

Nächste Schritte

- Benutzeranleitung
- Realisierungsbericht
- Und noch einmal die ganze Dokumentation durchlesen

Tabelle 18 Arbeitsjournal Tag 9

8.10. Tag 10: 23.05.2022

Tätigkeiten	Beteiligte Personen	Aufwand geplant (Std)	Aufwand effektiv (Std)
Schlussbericht	Yusup Khasbulatov	4	3
Projektabnahme (Daily Meeting)	Yusup Khasbulatov Ramun Hofmann Florian Baumgartner	0.25	0.5
Projektabschluss	Yusup Khasbulatov Ramun Hofmann Florian Baumgartner	1	1
Reserve	Yusup Khasbulatov	1	0.75
Benutzerhandbuch	Yusup Khasbulatov	0	1
Korrekturlesen	Yusup Khasbulatov	0.25	0.5
Arbeitsjournal; Backup	Yusup Khasbulatov	1.5	0.75
Total:		8	7.5

Tagesablauf

Heute habe ich die Dokumentation ein paar Mal durchgelesen. Die Aufgabe "Abschlussbericht" war in dem Sinne auch ein Korrekturlesen, so dass ich hier weniger Aufgaben schreiben konnte. Ich habe alle fehlenden Informationen ergänzt und ein paar Dinge getan, die zur Realisierungsphase gehören. Ich habe den ganzen Tag nur mit Schreiben und Korrigieren und Ergänzen verbracht. Ich habe die Dokumentation auch ein paar Mal in PDF umgewandelt, um sicherzustellen, dass alles korrekt ist.

Wir haben die Projektabnahme und den Projektabschluss gemeinsam durchgeführt. Meine Vorgesetzten wollten auch, dass ich mehr an dem Dokument arbeite und es zum Endpunkt bringe, also war es sehr kurz, die restliche Zeit in der Aufgabe Projektabschluss verbrachte ich damit, die Sitzungsprotokolle auszufüllen und die Screenshots der Phasenfreigaben zu erstellen.

Hilfestellung

Heute musste ich nicht viel recherchieren, ich habe kurz nachgeschaut, wie man ein "continue" im Sequenzdiagramm darstellt und habe nichts dazu gefunden, also habe ich einen Selbstverweis in das Diagramm eingefügt, der erklärt, dass die Schleife von vorne beginnen soll.

Ausserdem hat mir Herr Guggisberg heute beim Korrekturlesen geholfen und mir die deutschen Grammatikfehler gezeigt.

Reflexion

Positiv

Das Exportieren funktionierte bei mir heute ohne Probleme.

Negativ

Heute habe ich keine negativen Punkte zu berichten.

Erkenntnisse

Zum Exportieren kann ich Windows Virtual Maschine verwenden, da es unter Windows gut funktioniert.

Nächste Schritte

Präsentation vorbereiten

Tabelle 19 Arbeitsjournal Tag 10

9. Abschlussbericht

9.1. Vergleich Ist/Soll

Alle gestellten Aufgaben wurden erfüllt. Das aktualisierte System zeigt Wetterdaten, die aus öffentlichen Quellen abgerufen werden. Das Gerät selbst wurde nur leicht verändert, aber die Kunststoffsäulen, die jetzt aus einem undurchsichtigen Kunststoff bestehen, sind augenfällig.

Das ausgewählte Projekt wurde wie geplant innerhalb des vorgegebenen Zeitrahmens durchgeführt. Bereits nach Fertigstellung des Projekts wurden bei der Abnahme des Projekts einige Verbesserungen vorgeschlagen, die nicht in diese Umsetzung einbezogen wurden, aber in Zukunft umgesetzt werden könnten (weitere Details sind in den Sitzungsprotokollen zu finden).

9.2. Persönliches Fazit

Während der Prüfung habe ich verschiedene Phasen durchlebt, von angenehm bis sehr stressig. Es besteht das Gefühl, dass es langfristig keine grösseren Schwierigkeiten geben wird als dieser Prüfung, aber das ist eine naive Behauptung.

Die Ernsthaftigkeit dieses Projekts in Bezug auf mein zukünftiges Leben hat mich angetrieben und motiviert. Ich habe bei diesem Projekt nicht nach einfachen Wegen gesucht. Zum Beispiel hätte ich die Möglichkeit gehabt, fertige Vorlagen für die Dokumentation zu verwenden, aber ich habe mich entschieden, meine eigenen zu erstellen. Dennoch sind die Vorlagen, die mir die GIBB-Schule zur Verfügung gestellt hat, sehr nützlich, da sie viele Kommentare zu bestimmten Themen enthalten.

In der Programmiersprache Python habe ich keine neuen, völlig unbekannten Themen gelernt. Denn ich verwende Python ziemlich häufig in meinen Hobbyprojekten und dies hat mir geholfen, die Implementierungsphase in kurzer Zeit abzuschliessen. Der Zweck dieses Projekts bestand jedoch auch nicht darin, etwas komplett Neues zu lernen, sondern bestehendes Wissen in der Praxis anzuwenden. Trotzdem erlebte ich Überraschungen: Natürlich kamen mir einige Details, wie z.B. die Beziehungen von Objekten zueinander im Django Framework, erst beim Erstellen von UML-Diagrammen in den Sinn.

9.3. Schlussreflexion

Diese Arbeit war sehr intensiv, die Dokumentation ist das bisher umfangreichste Dokument, das ich in meinem Leben geschrieben habe. Damit habe ich mir eine solide Grundlage geschaffen und viele, die in dieser Dokumentation angewandten Techniken, werde ich bei künftigen Projekten einsetzen.

Teil 2: Projektdokumentation

Projektname: Wackeldisplay Wetter

Autor: Yusup Khasbulatov



10. Einleitung

Wer ist 89grad GmbH?

Die Geschichte von 89grad begann 2010, als sich die Gründer Ramun und Florian von ihrer jahrelangen Arbeit in der perfekten, rechteckigen, standardisierten Welt der grossen Unternehmen verabschiedeten, um ihre Ziele zu verfolgen: Prototypen und Produkte schnell und einfach auf den Markt zu bringen und sie mit echtem Marktfeedback zu verifizieren. Daher auch die 89 Grad im Firmennamen als Hinweis darauf, dass nicht alles perfekt und rechtwinklig, wie ein 90-Grad-Winkel sein muss.

Getreu dem Motto «weniger pitchen, mehr umsetzen» gründeten sie 89grad und entwickeln seither kundenorientierte Lösungen in der digitalen Welt. Die Gründung des FabLab Bern führte auch zu einer eigenen Hardware-Entwicklungswerkstatt.

Mit einem Team von engagierten Spezialisten entwickelt 89grad heute innovative Hardware- und Softwarereprodukte.

Arbeitsbereich und Aufgabenstellung

Individuelle Entwicklung - Der gesamte Prozess vom Requirements Engineering, über die Planung der Softwarearchitektur bis hin zur Entwicklung der Anwendung wird auf die persönlichen Bedürfnisse des Kunden zugeschnitten.

Verbindung Open und Closed Source Software - Wo möglich, setzt 89grad GmbH Open Source Komponenten ein, damit die Anwendungen der Kunden einfach und kostengünstig zu erstellen und zu betreiben sind. Einfache Bedienbarkeit, Lean Design und Sicherheit sind integrale Bestandteile des Software-Entwicklungsprozesses der 89grad GmbH.

Vielfältige Integrationen - 89grad GmbH hat Erfahrung darin, Legacy-Systeme mit neuen IT-Landschaften zu verbinden und die Systeme mit modernen Web-APIs zu erweitern. Typische Systeme, mit denen wir arbeiten, sind ERP (wie Salesforce, Axapta, Dynamics) oder periphere Systeme (wie Mail, Security, Firewalls).

Agiles Projektmanagement - Um Entwicklungszyklen so kurz wie möglich zu halten, die Ideen und Systeme für die schnelle und direkte Integration in den Markt zu evaluieren, setzt 89grad GmbH auf Verfahren aus dem Silicon Valley. Durch mehrere Besuche im Silicon Valley ist 89grad GmbH mit den Informationsmethoden und Konzepten bestens vertraut.

Diverse Programmiersprachen - 89grad GmbH setzt hauptsächlich Python und Django ein, verfügt aber auch über langjährige Erfahrung in der Anwendung anderer Sprachen wie C++, JavaScript, HTML und PHP.

Hosting und Support - Wenn es von Kunden gewünscht wird, kann die 89grad GmbH auch das Hosting nach der Erstellung der Applikation übernehmen. 89grad GmbH bevorzugt Schweizer Provider und Cloud Services, so dass die Daten der Kunden geschützt in der Schweiz liegen.

Weshalb wurde das Projekt/IPA im gewählten Themenbereich initialisiert?

Die Hardware des Wackeldisplays wurde als «Proof of Concept» hergestellt und diente als Beweis, um zu zeigen, dass 89grad GmbH in der Lage ist, solche oder ähnliche Geräte

herzustellen. In Diskussionen stellten wir in der Folge fest, dass die Anzeige der Abfahrtszeiten öffentlicher Verkehrsmittel für dieses Gerät nicht ganz geeignet ist und wir beschlossen, das Format zu ändern. Wir suchten also einen Datensatz, der nicht zeitkritisch ist, und so kamen wir auf die Wetterdaten. Nun sollte dieses Display statt der Abfahrtszeiten der öffentlichen Verkehrsmittel die Wettertendenz anzeigen.

Was ist der Mehrwert dieser Arbeit?

Wie oben bereits erwähnt wurde, handelt es sich um eine Art Werbung für das Know-how und die Möglichkeiten, die die 89grad GmbH anbieten kann. Bisher wurde noch keine genaue Messung des Feedbacks für dieses Gerät durchgeführt und wir sind in der Phase, in der wir herausfinden wollen, welche Datensätze für eine solche Hardware geeignet sind. Ziel dieser Arbeit ist es, zu überprüfen, wie einfach es ist, das Format der Daten zu ändern. Dies dient als Grundlage für zukünftige Entwicklungen. Zudem wollen wir prüfen, inwieweit die Integration von Webanwendungen für dieses Projekt geeignet ist.³

³ Der Text in dem Kapitel Einleitung ist eine leicht von mir angepasste Version, die auf 89grad GmbH Webseite unter folgendem Link zu finden ist: <https://www.89grad.ch/softwareloesungen-kundenportale>. Zudem wurde es auch im [Arbeitsjournal](#) erwähnt.

11. Initialisierung

Gemäss HERMES-gibb wird in der Initialisierungsphase die bestehende Situation detailliert beschrieben. Das Ziel dieser Phase ist, die Ausgangslage für das Projekt zu definieren und die Projektziele zu vereinbaren. Dieser Teil wird als Grundlage für die weiteren geplanten Phasen dienen.

11.1. Studie; Ist-Zustand

In diesem Kapitel werden die verschiedenen Schnittstellen beschrieben und der aktuelle Stand betrachtet.

11.1.1. Systemarchitektur

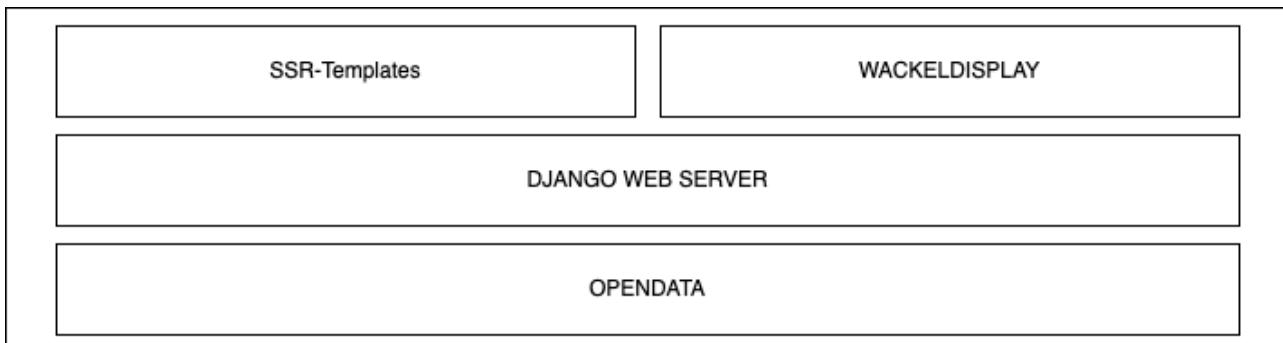


Abbildung 16 IST-Zustand: Systemarchitektur

Die Django Web-Server Schnittstelle holt die Daten vom <https://opendata.ch> und passt sie auf die Templates an. Die UI-Schnittstelle wird über SSR erstellt, Django setzt dafür jinja-Template Engine ein. Die Daten werden dann in einer Tabelle dargestellt. Für die Darstellung der Daten und deren **Filterung** werden AJAX-Anfragen eingesetzt. Die Wackeldisplay-Schnittstelle (Hardware) wird aufgerufen, indem ein QR-Code gescannt wird. Die Daten werden dann Schritt für Schritt angezeigt.

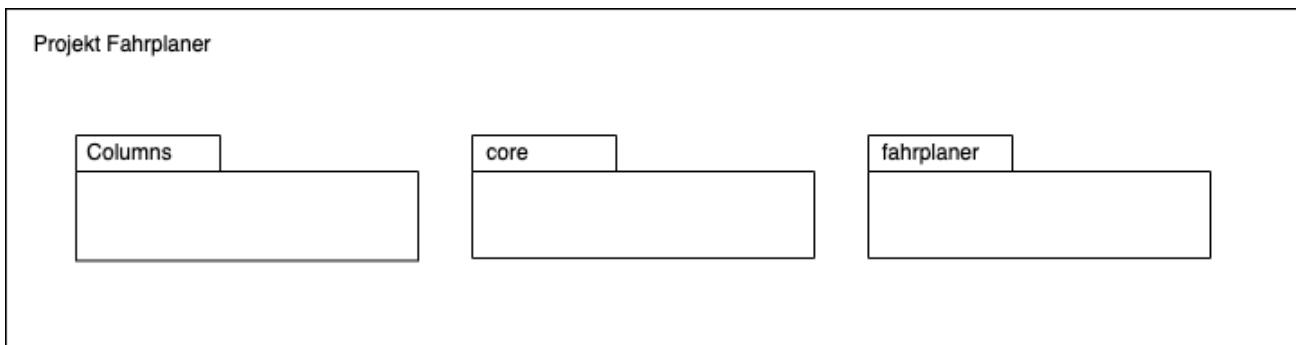


Abbildung 17 IST-Zustand: Django Projekt Architektur

Paket	Beschreibung
Columns	Dieses Paket ist zuständig für die Konfiguration des Wackeldisplays.
core	Dieses Paket wird automatisch von «django-admin» CLI während der Projekterstellung generiert. Es wird für die Speicherung aller Projektdateien und Konfigurationen verwendet.

fahrplaner	Dies ist ein Paket, das mit «django-admin» als Anwendung erstellt wurde (Django bezeichnet die zusätzlichen Pakete als «application»). Hier ist die Business Logik implementiert für die Darstellung, Verwaltung und Steuerung der Daten.
------------	---

Tabelle 20 IST-Zustand: Django Projekt Beschreibung

11.1.2. Wackeldisplay (Fahrplaner)

Das Wackeldisplay ist eine Hardware, welche aus 5 Spalten besteht. Jede Spalte wird mit einem Schrittmotor angesteuert. Ein Teil des Wackeldisplays wurde bereits in der [Kurzfassung des IPA Berichtes](#) beschrieben. An dieser Stelle wiederhole ich die wichtigsten Punkte, um den Kontext herzustellen.

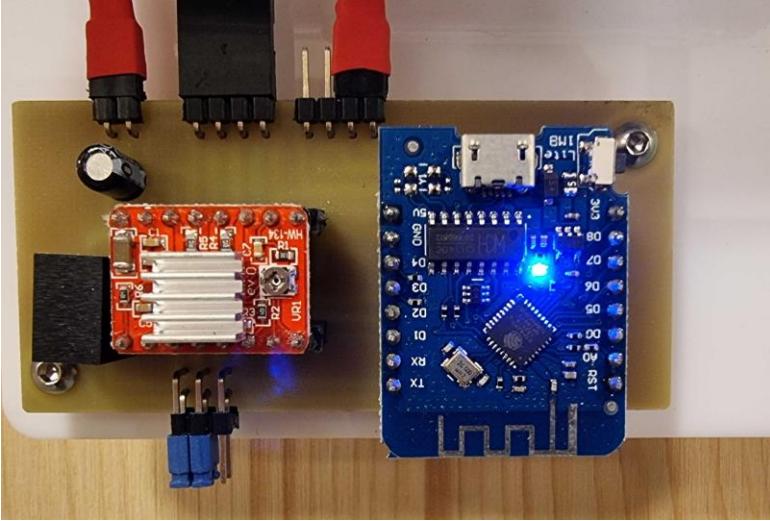
Bild	Beschreibung
	Ansicht: Wackeldisplay von vorne. Man sieht 5 transparente Spalten, auf welche in gleichen Abschnitten Text oder Symbole geklebt sind. Auf die linke Seite jeder Spalte ist der Schrittmotor zu erkennen, welcher die Spalte nach oben oder nach unten bewegt. Im obersten Teil der Spalte ist die Beschreibung der Information enthalten, welche in der Spalte angezeigt wird.
	Der Schrittmotor wird über WLAN gesteuert. Für diese Funktionalität wird ein ESP8266-Mikrocontroller verwendet. Auf dem ESP8266 läuft eine Software, die sich mittels des MQTT-Protokolls mit dem Server austauscht. Jeder ESP8266 Microcontroller hört lediglich auf den Befehl auf welche Position er den Schrittmotor steuern soll.

Tabelle 21 IST-Zustand: Wackeldisplay Beschreibung

11.1.3. MQTT-Server

Die 5 ESP8266-Microcontroller kommunizieren mit dem Django-Server über das WLAN.

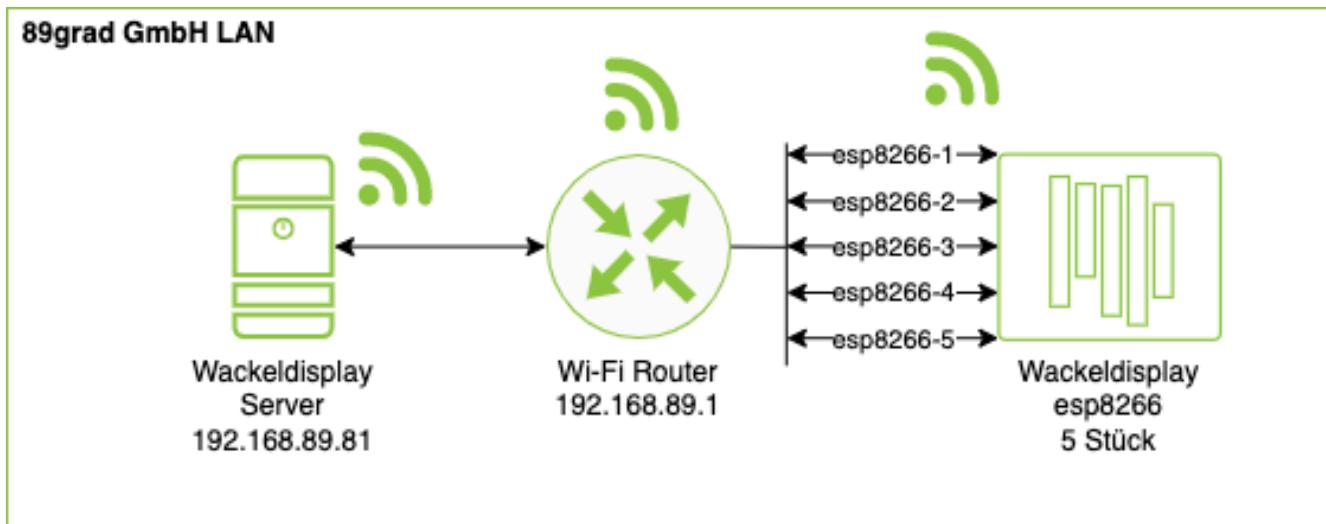


Abbildung 18 IST-Zustand: Wackeldisplay Kommunikation mit Django Server

11.1.4. Datenbank (SQLite)

Django Server verwendet eine SQLite-Datenbank, die während des Migrationsbefehls automatisch direkt unter dem Projekt Root-Pfad angelegt wird.

```
fahrplan@fahrplan:~/fahrplaner$ ls -l
total 192
drwxr-xr-x 5 fahrplan fahrplan 4096 Apr 11 06:13 Columns
drwxr-xr-x 2 fahrplan fahrplan 4096 Nov 18 05:08 certs
drwxr-xr-x 3 fahrplan fahrplan 4096 Nov 22 08:42 core
-rw-r--r-- 1 fahrplan fahrplan 155648 Apr 12 08:03 db.sqlite3
drwxr-xr-x 7 fahrplan fahrplan 4096 Apr 11 06:13 fahrplaner
drwxr-xr-x 2 fahrplan fahrplan 4096 Nov 18 05:08 logs
-rw-r--r-- 1 fahrplan fahrplan 660 Nov 18 04:59 manage.py
-rw-r--r-- 1 fahrplan fahrplan 48 Nov 18 04:59 requirements.txt
drwxr-xr-x 3 fahrplan fahrplan 4096 Nov 22 08:43 staticfiles
drwxr-xr-x 2 fahrplan fahrplan 4096 May  5 10:47 tmp
fahrplan@fahrplan:~/fahrplaner$
```

Abbildung 19 IST-Zustand: SQLite Datenbank Datei

Die ERD-Grafik zeigt die bestehenden Tabellen, die lediglich zum Abspeichern der Konfiguration dienen.

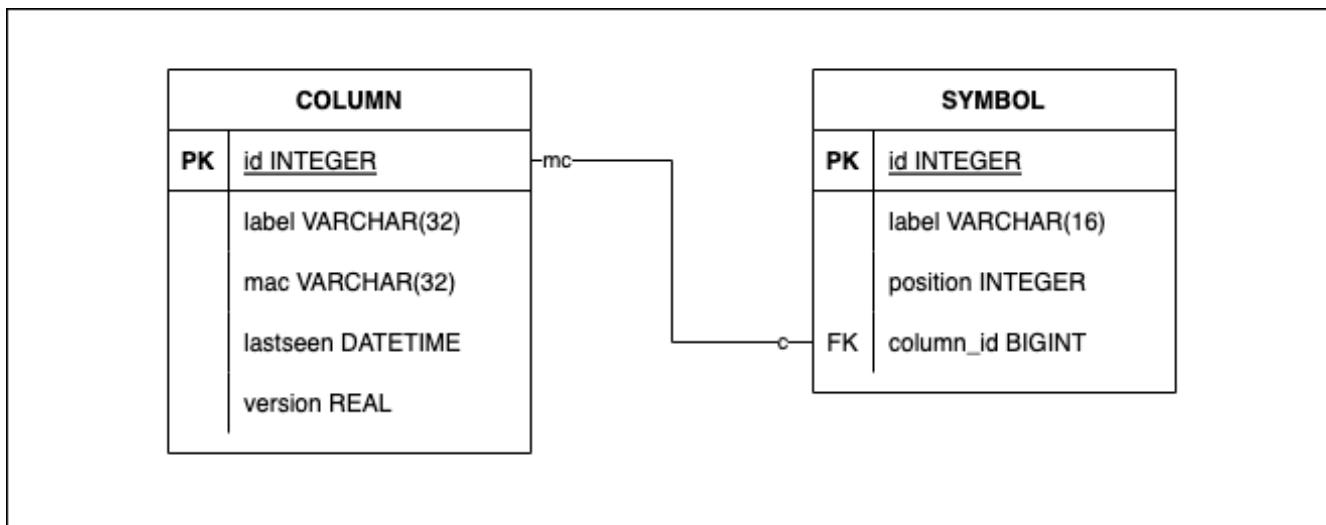


Abbildung 20 IST-Zustand: Wackeldisplay ERD

Tabelle	Beschreibung
Column	Diese Entität entspricht der Spalte auf dem Wackeldisplay, daher sollte es so viele Einträge geben, wie es Spalten auf dem Wackeldisplay gibt. Hier wird eine menschenlesbare Bezeichnung erstellt (soll jedoch Manuel erstellt werden) und die MAC-Adresse des esp8266 gespeichert. Auch das Datum der letzten Aktualisierung wird gespeichert.
Symbol	Diese Einheit entspricht den Symbolen, die in jeder Spalte vorhanden sind. Sie hat auch eine menschenlesbare Bezeichnung, eine Beziehung zur Spaltentabelle und die aktuelle Position der Spalte.

Tabelle 22 IST-Zustand: Datenbank Tabellen Beschreibung

Die Beziehung wird in modifizierter Chen-Notation⁴ geschrieben. Für jedes Symbol sowie jede Spalte wird ein Eintrag in der Datenbank (oder keine) erstellt. Da eine Spalte aus mehreren Symbolen besteht, haben wir hier eine Foreign Key Beziehung von Symbolen zur Spaltentabelle.

Und hier ist ein Screenshot von der physischen Datenbank:

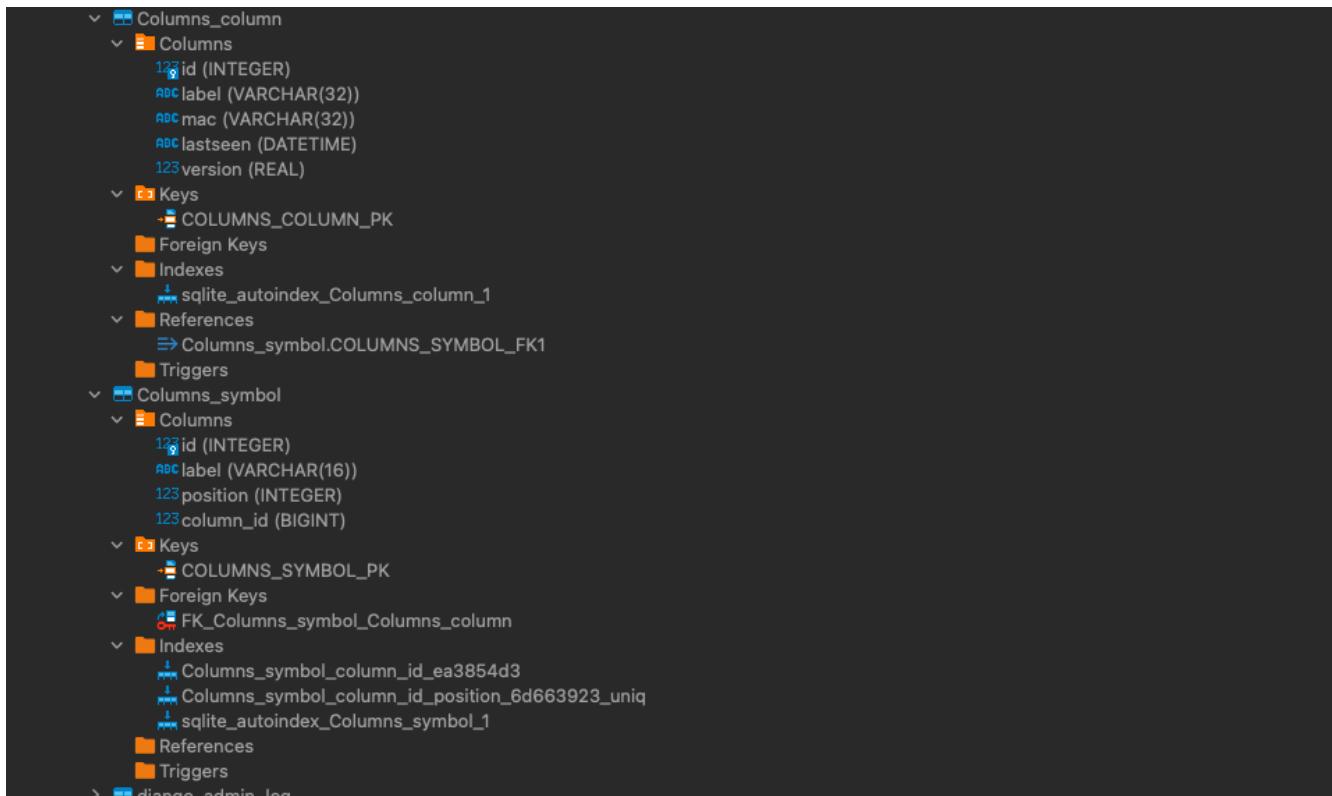


Abbildung 21 IST-Zustand: Physischer Datenbankzustand

11.1.5. MVC Komponente

Eine statische HTML-Seite weiss nicht, wie sie auf Benutzeraktionen reagieren soll. Für eine Zwei-Seitige-Interaktion sind dynamische Webseiten erforderlich.

MVC steht für Model-View-Controller. Dabei handelt es sich um eine Methode zur Organisation von Programmcode, wobei verschiedene Blöcke, die für unterschiedliche Aufgaben zuständig

⁴ [https://de.wikipedia.org/wiki/Chen-Notation#Modifizierte_Chen-Notation_\(MC-Notation\)](https://de.wikipedia.org/wiki/Chen-Notation#Modifizierte_Chen-Notation_(MC-Notation))

sind, voneinander getrennt werden. Der eine Block ist für die Anwendungsdaten zuständig, der andere für das Aussehen, und der dritte Block steuert die Anwendung.

Das Backend wurde auf dem Django Web Framework entwickelt. Django benennt das MVC-Pattern ein wenig anders, obwohl es schlussendlich aufs Gleiche hinausläuft. Sie nennen es MTV. Also anstatt Model-View-Controller Model-Template-View:

MVC	MTV
Model	Model
View	Template
Controller	View

Tabelle 23 IST-Zustand: MVC und MTV Vergleich

Model - diese Komponente ist für die Daten zuständig und definiert auch die Struktur der Anwendung. In diesem Projekt übernimmt Model Komponente eine Mapping zu der Datenbank. Da ich das Django Web Framework verwende, erleichtert es mir die Datenverarbeitung, da Django dank des ORM-Frameworks die gesamte Validierung der ankommenden Daten automatisch durchführen kann.

Template (View) - diese Komponente ist für die Interaktion mit dem Benutzer zuständig. Das heißt, der Code der Template Komponente definiert das Aussehen und die Verwendung der Anwendung. In diesem Projekt ist ein interaktives Element geplant, mit dem der Benutzer bestimmte Daten auf dem Wackeldisplay anzeigen kann.

View (Controller) - diese Komponente ist für die Kommunikation zwischen Modell und Ansicht zuständig. Der View Code definiert, wie die Webseite mit dem Benutzer interagiert. Dies ist im Wesentlichen das Gehirn der MVC-Anwendung.

Das Wackeldisplay Modelklassen Diagramm

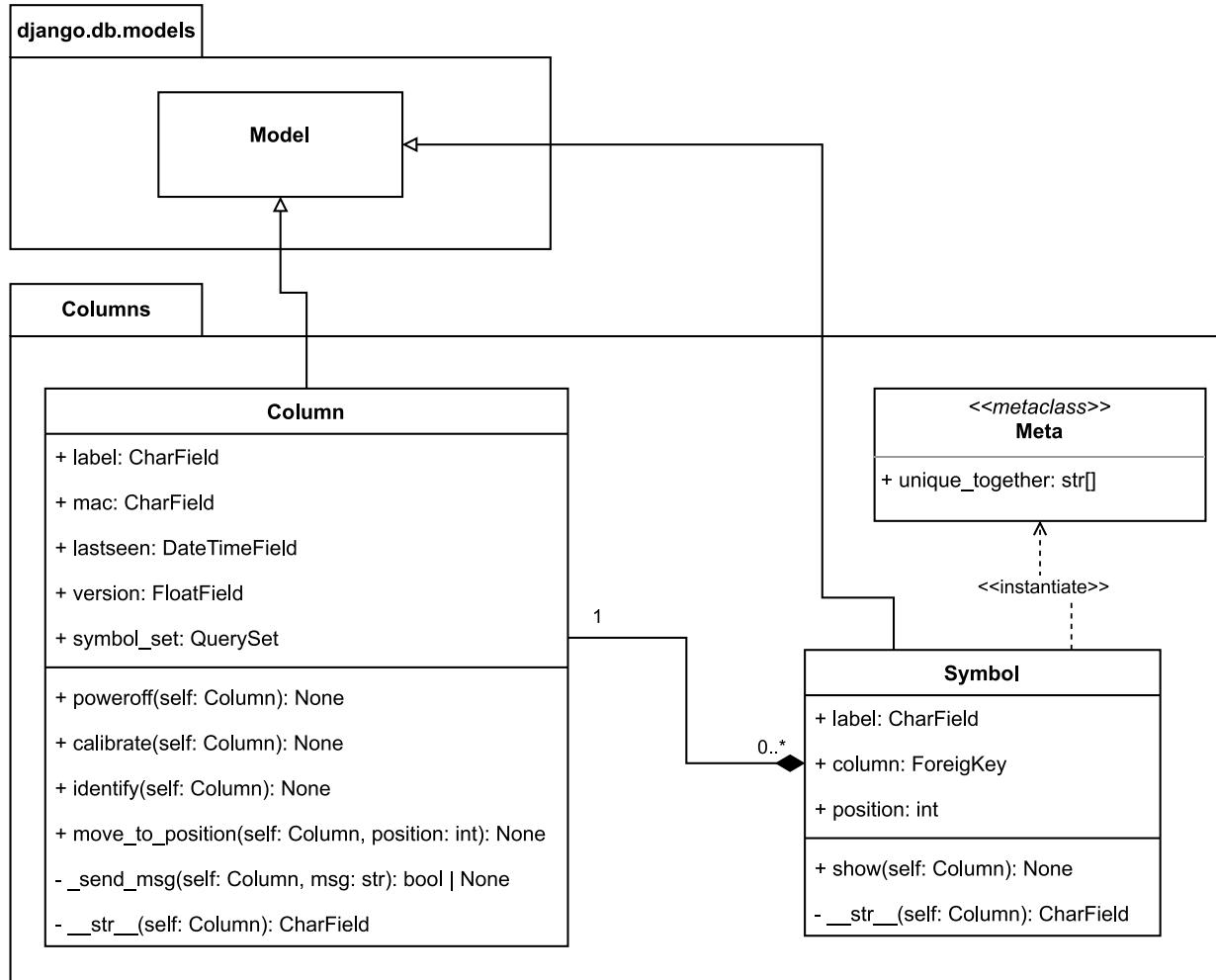


Abbildung 22 IST-Zustand: Klassendiagramm

Das Diagramm besteht aus zwei Klassen, die dem Datenmodell entsprechen. Beide Klassen werden von der Django Model-Klasse abgeleitet. Die Django Model-Klasse⁵ verfügt über ein ORM, welches für die Kommunikation und Modifikation der Datenbank verwendet werden kann. Die Meta-Klasse, welche in die Symbol-Klasse instanziiert wird, wird benötigt, um die «Unique Constraint»⁶ in der Datenbank zwischen «column» und «position» Attribute zu erstellen. Dies wird gemacht, um sicher zu stellen, dass nicht gleiche Positionen für zwei Symbole generiert werden. Die beiden Klassen stehen in einer Kompositionsbeziehung zueinander, d. h. die Lebensdauer der Symbolobjekte hängt von der Lebensdauer der Spaltenobjekte ab. Die Multiplizität dieser Komposition ist wie folgt: eine Column kann 0 oder mehr Symbole enthalten. Ein Symbol gehört zu genau einer Column.

Wackeldisplay Templates (View)

Wenn man den QR-Code ([siehe Abbildung oben](#)) mit dem Handy scannt, wird die URL <http://fahrplaner.e12> geöffnet. Die Root URL ist so konfiguriert, dass es automatisch den Browser auf <http://fahrplaner.e12/fahplaner> weiterleitet. Diese wird weiter unten noch, genauer beschrieben. Daraufhin wird folgendes Template geladen (Screenshot wurde auf der lokalen

⁵ <https://docs.djangoproject.com/en/4.0/ref/models/instances/#django.db.models.Model>

⁶ <https://docs.djangoproject.com/en/4.0/ref/models/options/#unique-together>

«Development» Umgebung gemacht. <http://127.0.0.1:8000/fahrplaner/> entspricht somit <http://fahrplaner.e12/fahplaner>):

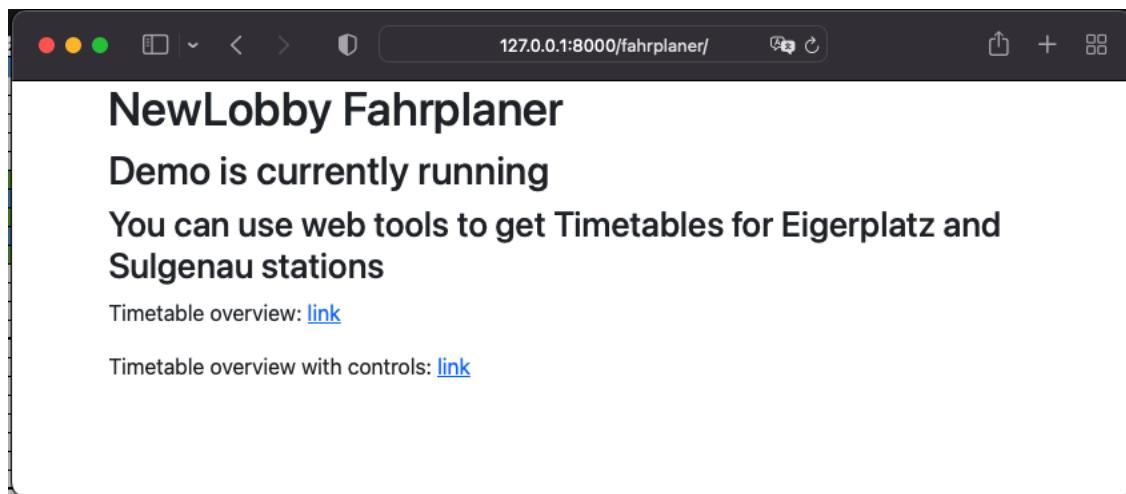


Abbildung 23 IST-Zustand: Fahrplaner Home

Je nach Situation wird hier verschiedener Text angezeigt. Wenn die Demo bereits läuft, wird «Demo is currently running» angezeigt, sonst «Demo is started». Die Links unten («Timetable overview» und «Timetable overview with controls») sind nur im ersten Fall ersichtlich sein.

Screenshot	Beschreibung																																																												
A screenshot of the 'NewLobby Fahrplaner' visual page. The URL in the address bar is '127.0.0.1:8000/fahrplaner/visual/'. The page displays two tables of bus departure times. The first table is for 'Eigerplatz' and the second for 'Sulgenau'. Both tables have columns for Transport (Bus or Tram), Direction, Departure in, and Number. Eigerplatz <table border="1"><thead><tr><th>Transport</th><th>Direction</th><th>Departure in</th><th>Number</th></tr></thead><tbody><tr><td>Bus</td><td>Bern, Bahnhof</td><td>3</td><td>10</td></tr><tr><td>Bus</td><td>Bern, Bahnhof</td><td>3</td><td>19</td></tr><tr><td>Tram</td><td>Bern Wankdorf, Bahnhof</td><td>6</td><td>9</td></tr><tr><td>Bus</td><td>Bern Wankdorf, Bahnhof</td><td>9</td><td>28</td></tr><tr><td>Bus</td><td>Bern, Bahnhof</td><td>12</td><td>10</td></tr><tr><td>Bus</td><td>Bern, Bahnhof</td><td>12</td><td>19</td></tr><tr><td>Tram</td><td>Bern Wankdorf, Bahnhof</td><td>12</td><td>9</td></tr><tr><td>Bus</td><td>Bern, Weissenbühl</td><td>12</td><td>28</td></tr></tbody></table> Sulgenau <table border="1"><thead><tr><th>Transport</th><th>Direction</th><th>Departure in</th><th>Number</th></tr></thead><tbody><tr><td>Bus</td><td>Bern, Weissenbühl</td><td>3</td><td>19</td></tr><tr><td>Bus</td><td>Bern, Bahnhof</td><td>9</td><td>19</td></tr><tr><td>Bus</td><td>Bern, Weissenbühl</td><td>12</td><td>19</td></tr><tr><td>Bus</td><td>Bern Wankdorf, Bahnhof</td><td>12</td><td>28</td></tr><tr><td>Bus</td><td>Bern, Weissenbühl</td><td>12</td><td>28</td></tr></tbody></table>	Transport	Direction	Departure in	Number	Bus	Bern, Bahnhof	3	10	Bus	Bern, Bahnhof	3	19	Tram	Bern Wankdorf, Bahnhof	6	9	Bus	Bern Wankdorf, Bahnhof	9	28	Bus	Bern, Bahnhof	12	10	Bus	Bern, Bahnhof	12	19	Tram	Bern Wankdorf, Bahnhof	12	9	Bus	Bern, Weissenbühl	12	28	Transport	Direction	Departure in	Number	Bus	Bern, Weissenbühl	3	19	Bus	Bern, Bahnhof	9	19	Bus	Bern, Weissenbühl	12	19	Bus	Bern Wankdorf, Bahnhof	12	28	Bus	Bern, Weissenbühl	12	28	Der erste Link führt zur Übersicht der geladenen Daten, die das Wackeldisplay momentan anzeigt.
Transport	Direction	Departure in	Number																																																										
Bus	Bern, Bahnhof	3	10																																																										
Bus	Bern, Bahnhof	3	19																																																										
Tram	Bern Wankdorf, Bahnhof	6	9																																																										
Bus	Bern Wankdorf, Bahnhof	9	28																																																										
Bus	Bern, Bahnhof	12	10																																																										
Bus	Bern, Bahnhof	12	19																																																										
Tram	Bern Wankdorf, Bahnhof	12	9																																																										
Bus	Bern, Weissenbühl	12	28																																																										
Transport	Direction	Departure in	Number																																																										
Bus	Bern, Weissenbühl	3	19																																																										
Bus	Bern, Bahnhof	9	19																																																										
Bus	Bern, Weissenbühl	12	19																																																										
Bus	Bern Wankdorf, Bahnhof	12	28																																																										
Bus	Bern, Weissenbühl	12	28																																																										

The screenshot shows a web browser window with the URL 127.0.0.1:8000/fahrplaner/controls/. The title bar says "NewLobby Fahrplaner". Below it, a message says "Choose on the Direction button to get Timetable". There are two rows of buttons for selecting transport modes and stations. The first row includes "Bern, Bahnhof", "Bern, Weissenbühl", "Bern Wankdorf, Bahnhof", and "Köniz, Schloss". The second row includes "Ostermundigen, Rüti", "Bern, Elfenau", and "Bern Europaplatz, Bahnhof". Below these are two tables. The top table has columns: Transport, Station, Direction, Departure in, and Number. It lists bus departures from Eigerplatz to Bern Bahnhof at various times. The bottom table has columns: Transport, Station, Direction, Departure in, and Number. It lists bus departures from Sulgenau to Bern Bahnhof at various times.

Tabelle 24 IST-Zustand: Fahrplaner Screenshots

Wackeldisplay Views (Controller)

Das Wackeldisplay System unterstützt momentan folgende Anwendungsfälle:

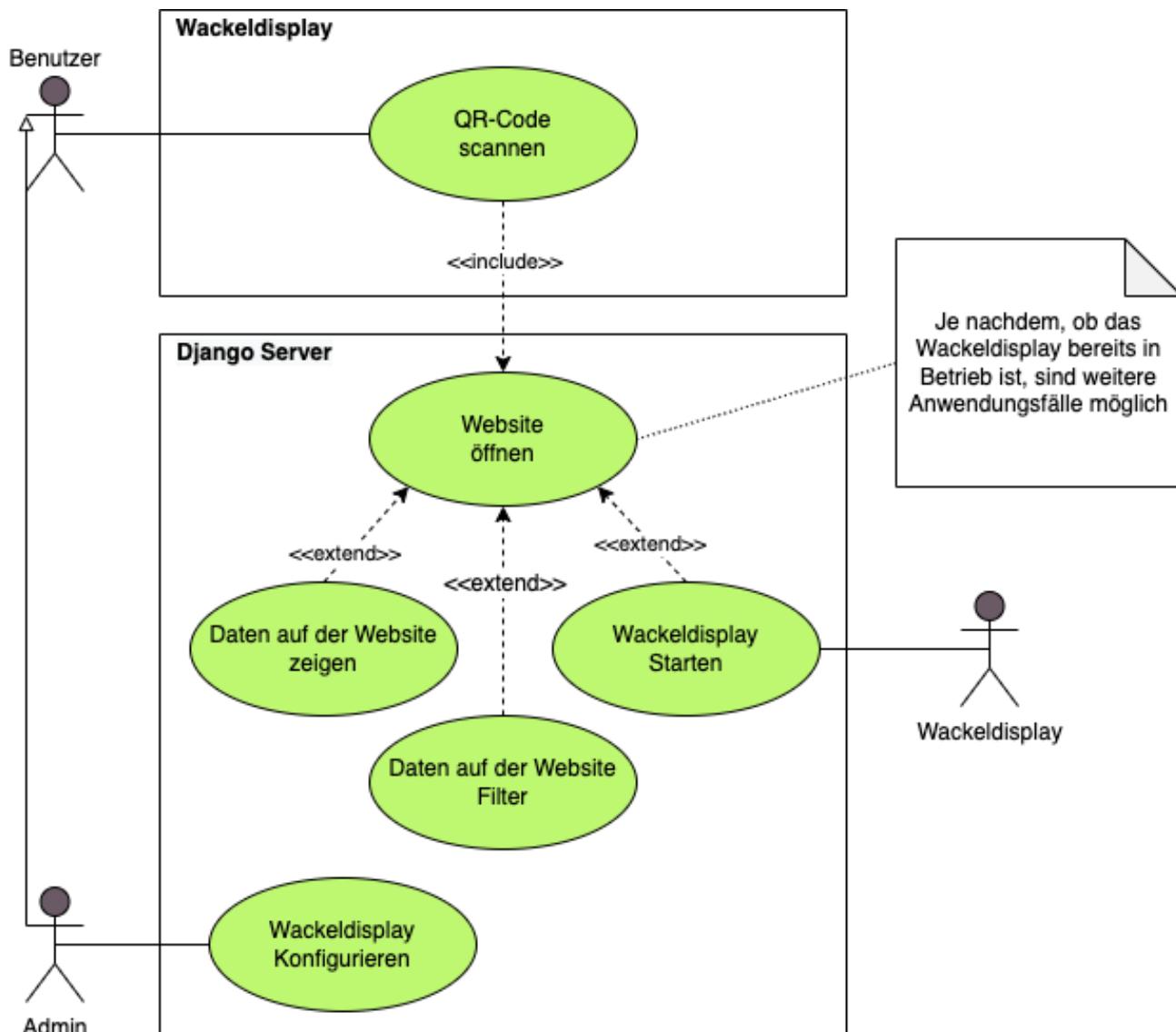


Abbildung 24 IST-Zustand: Django Server Anwendungsfälle

Anwendungsfall «QR-Code scannen»	
Kurzbeschreibung	Der Anwender/in scannt der QR-Code mit dem Handy.
Akteure	Jeder
Vorbedingungen	<ul style="list-style-type: none"> - Ein QR-Code muss vorhanden sein. - Der Anwender/in muss sich mit 89grad GmbH WiFi verbinden.
Ablauf	<ul style="list-style-type: none"> - Der Anwender/in scannt den QR-Code mit einem Handy.
Resultat	Das Handy erkennt die URL, welche im QR-Code enthalten ist.
Ausnahmen	Keine.
Anwendungsfall «Webseite öffnen»	
Kurzbeschreibung	Der Anwender/in scannt der QR-Code mit dem Handy und klickt auf den Link.
Akteure	Jeder
Vorbedingungen	<ul style="list-style-type: none"> - Ein QR-Code muss vorhanden sein. - Der Anwender/in muss sich mit dem 89grad GmbH WiFi verbinden. - Django Server muss gestartet sein.
Ablauf	<ul style="list-style-type: none"> - Der Anwender/in scannt den QR-Code mit einem Handy. - Der Anwender/in klickt auf den Link.
Resultat	Es wird eine Webseite geöffnet.
Ausnahmen	Keine.
Anwendungsfall «Wackeldisplay Starten»	
Kurzbeschreibung	Der Anwender/in startet der Wackeldisplay, in dem er/sie den QR-Code scannt.
Akteure	Jeder
Vorbedingungen	<ul style="list-style-type: none"> - Ein QR-Code muss vorhanden sein (ausgedrückt liegt neben oder drauf geklebt). - Das Wackeldisplay muss eingeschaltet sein. - Der Anwender/in muss sich mit dem 89grad GmbH WiFi verbinden. - Das Wackeldisplay Service (Worker) muss auf dem Server gestartet sein. - Der Django Server muss gestartet sein. - Das Wackeldisplay Konfiguration muss vorhanden sein.
Ablauf	<ul style="list-style-type: none"> - Der Anwender/in scannt den QR-Code mit einem Handy. - Der Anwender/in öffnet die URL. - Der Anwender/in wartet 1-10 Sekunden.
Resultat	<ul style="list-style-type: none"> - Das Wackeldisplay startet
Ausnahmen	Wenn das Wackeldisplay bereits vor kurzer Zeit gestartet wurde und daher nicht alle Bedingungen auf der Server Seite erfüllt sind, wird eine entsprechende Nachricht auf der Webseite angezeigt.

Anwendungsfall «Daten auf der Webseite zeigen»

Kurzbeschreibung	Der Anwender/in öffnet die Webseite, indem er/sie auf den Link «Timetable overview» klickt.
Akteure	Jeder
Vorbedingungen	<ul style="list-style-type: none"> - Ein QR-Code muss vorhanden sein (ausgedrückt liegt neben oder drauf geklebt). - Das Wackeldisplay muss eingeschaltet sein. - Der Anwender/in muss sich mit dem 89grad GmbH WiFi verbinden. - Das Wackeldisplay Service (Worker) muss auf dem Server gestartet sein. - Der Django Server muss gestartet sein. - Das Wackeldisplay Konfiguration muss vorhanden sein. - Das Wackeldisplay soll in Betrieb sein.
Ablauf	<ul style="list-style-type: none"> - Der Anwender/in scannt den QR-Code mit einem Handy. - Der Anwender/in öffnet die URL. - Der Anwender/in klickt auf den Link «Timetable overview» auf der Webseite.
Resultat	Die Seite mit einer Tabellenübersicht der angezeigten Daten wird geladen.
Ausnahmen	Wenn das Wackeldisplay nicht in Betrieb ist, wird der Link «Timetable overview» nicht angezeigt, sondern das Wackeldisplay wird gestartet und eine Rückmeldung auf der Webseite angezeigt.

Anwendungsfall «Daten auf der Webseite filtern»

Kurzbeschreibung	Der Anwender/in öffnet die Webseite, indem er/sie auf den Link «Timetable overview with controls» klickt.
Akteure	Jeder
Vorbedingungen	<ul style="list-style-type: none"> - Ein QR-Code muss vorhanden sein (ausgedrückt liegt neben oder drauf geklebt). - Das Wackeldisplay muss eingeschaltet sein. - Der Anwender/in muss sich mit dem 89grad GmbH WiFi verbinden. - Das Wackeldisplay Service (Worker) muss auf dem Server gestartet sein. - Der Django Server muss gestartet sein. - Das Wackeldisplay Konfiguration muss vorhanden sein. - Das Wackeldisplay soll in Betrieb sein.
Ablauf	<ul style="list-style-type: none"> - Der Anwender/in scannt den QR-Code mit einem Handy - Der Anwender/in öffnet das URL - Der Anwender/in klickt auf den Link «Timetable overview with controls» auf die Webseite

Resultat	Die Seite mit einer Tabellenübersicht der angezeigten Daten wird geladen. Über dieser Tabelle wird die Liste der verfügbaren Filter-Buttons aufgeführt.
Ausnahmen	Wenn das Wackeldisplay nicht in Betrieb ist, wird der Link «Timetable overview with controls» nicht angezeigt, sondern Wackeldisplay wird gestartet und eine Rückmeldung auf der Webseite angezeigt.
Anwendungsfall «Wackeldisplay konfigurieren»	
Kurzbeschreibung	Der Administrator/in öffnet den Django-Admin Bereich in dem er/sie die URL http://fahrplaner.e12/admin aufruft.
Akteure	Nur Administrator/in
Vorbedingungen	<ul style="list-style-type: none"> - Der Anwender/in muss sich mit dem 89grad GmbH WiFi verbinden. - Der Django Server muss gestartet sein.
Ablauf	<ul style="list-style-type: none"> - Der Administrator/in öffnet die URL http://fahrplaner.e12/admin - Der Administrator/in füllt das Login-Form aus. - Der Administrator/in klickt auf den «Login» Button.
Resultat	Der Django-Admin Bereich erscheint.
Ausnahmen	Der Benutzer muss als Administrator in der Datenbank definiert sein.

Tabelle 25 IST-Zustand Anwendungsfälle

Folgend ist das sequenzielle Diagramm aufgeführt, welches die Aktionen auf dem Server aufzeigt, wenn der QR-Code gescannt und die Webseite geöffnet wird. Da die Komponenten aus dem Django Web Framework zu komplex sind, stellt dieses Diagramm eine sehr starke Vereinfachung dar. Die Benennung der Objekte, Methoden und zurückgeschickten Daten sind nicht 1-zu-1 gleich wie im Django Web Framework.

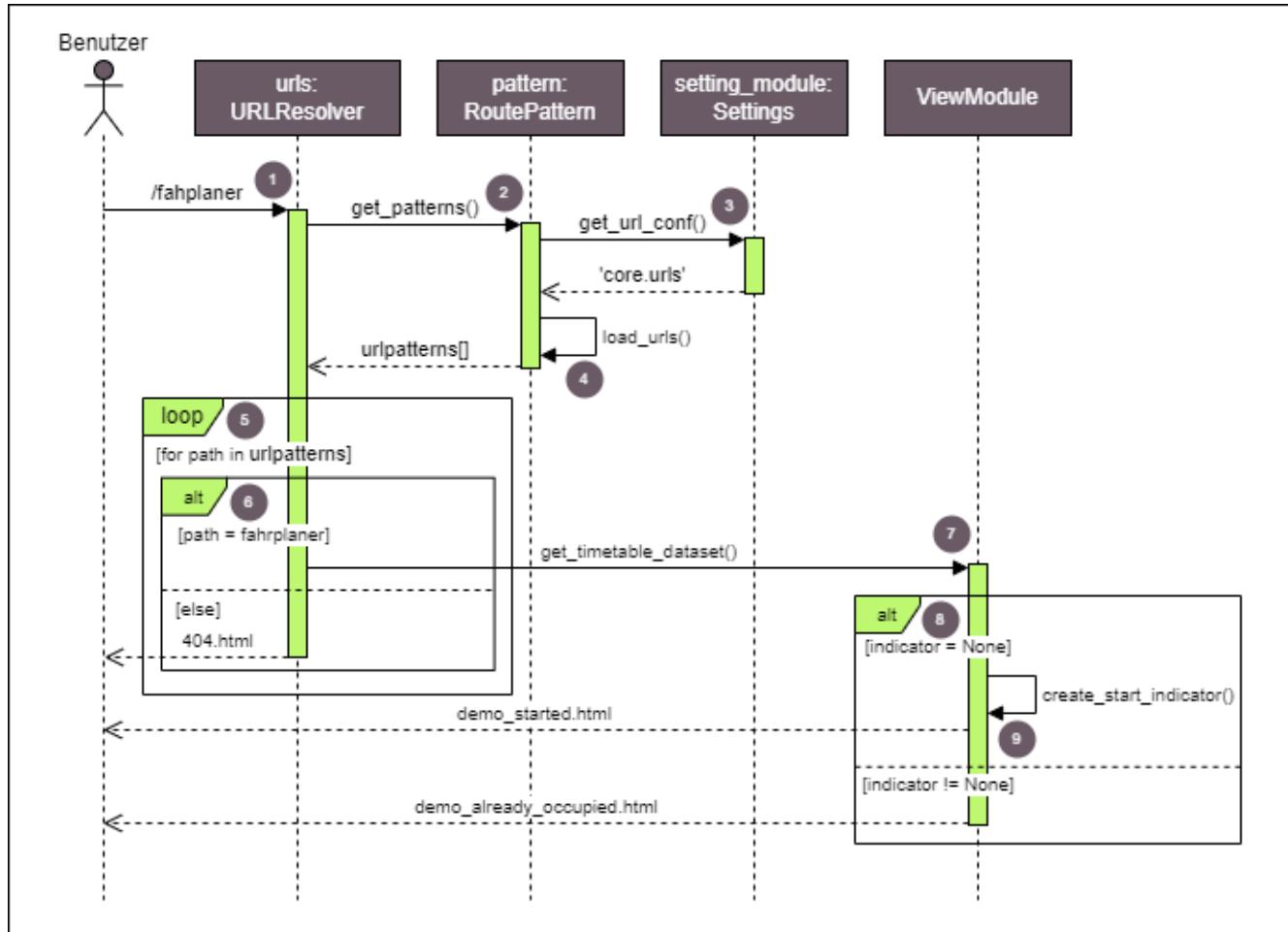


Abbildung 25 IST-Zustand: Sequenzdiagramm /fahrplaner Seite

Nr.	Beschreibung
1	Der Anwender/in scannt den QR-Code und öffnet die Webseite.
2	Django holt die bestehenden URL Pattern von den RoutePattern.
3	Das RoutePattern Objekt holt den Pfad zum URL Konfigurationsmodul vom Settings Modul.
4	Das RoutePattern-Objekt lädt die urls Module und schickt die definierte URL-Liste zurück.
5	Der URLResolver schleift über die URL-Liste bis er den Pfad «/fahrplaner» findet.
6	Falls dieser Pfad nicht existiert, schickt wird eine 404 Fehlermeldung zurückgeschickt.
7	Sonst ruft die Methode get_timetable_dataset() vom Views Module auf.

8	Die Routine überprüft, ob das Wackeldisplay bereits in Betrieb ist, indem sie nach den «Indikator» sucht. Wenn es diesen nicht gibt, startet das Wackeldisplay und lädt das Bestätigungs-Template.
9	Wenn es den Indikator gibt, wird das Template mit weiteren Links, die in Use Case Diagramm beschrieben sind, geladen.

Tabelle 26 IST-Zustand: /fahrplaner Seite Sequenzdiagramm Beschreibung

11.1.6. Routing

In der Aktuellen System sind folgende Routings (URLs) vorhanden

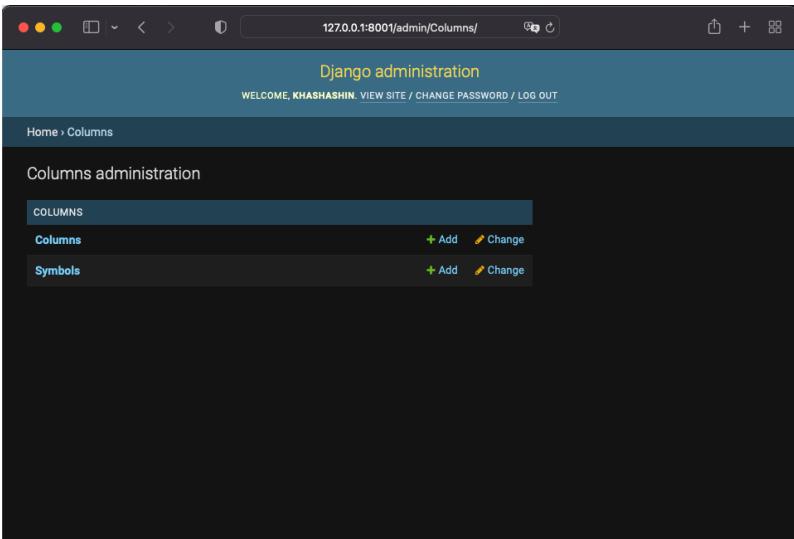
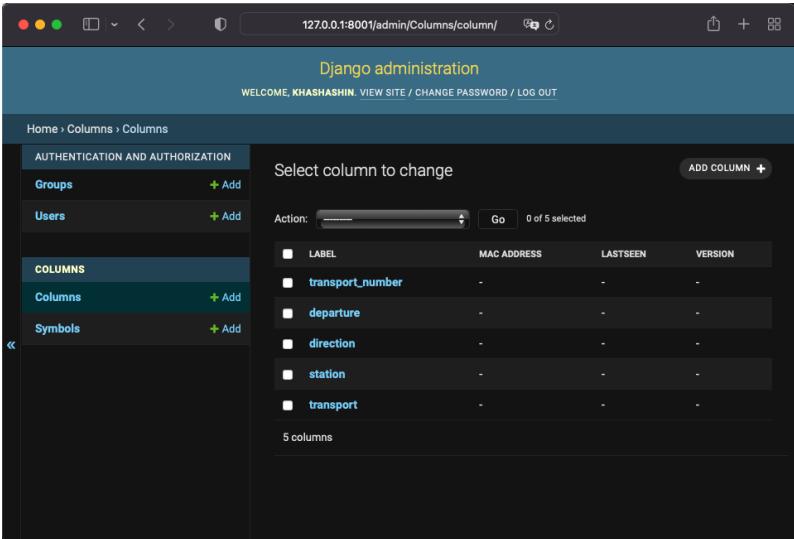
Route	Beschreibung
http://fahrplaner.e12/	Dieser Pfad wird an http://fahrplaner.e12/fahrplaner weitergeleitet
http://fahrplaner.e12/admin	Django Admin Bereich. Hier muss man sich einloggen, um die Daten verwalten zu können. Wird für Konfiguration der Wackeldisplay verwendet.
http://fahrplaner.e12/fahrplaner	Wird je nachdem zwei verschiedene Template-Blöcke gerendert. Wenn das Wackeldisplay bereits in Betrieb ist, wird ein Block mit weiteren Links gerendert, ansonsten wird eine Rückmeldung gerendert, dass das Wackeldisplay erfolgreich gestartet wurde.
http://fahrplaner.e12/fahrplaner/visual	Hier wird eine Tabelle mit Daten gerendert.
http://fahrplaner.e12/fahrplaner/controls	Hier wird eine Tabelle mit Daten gerendert und zusätzlich eine Filtermöglichkeit gegeben.

Tabelle 27 IST-Zustand: Vorhandene Routing

11.1.7. Django-Admin Bereich

Django verfügt über eine Datenverwaltungsfunktion «out of the box». Allerdings benötigt es eine gewisse Konfiguration im Backend. Man kann bestehende Modelle in einem Web darstellen und Django stellt alle CRUD-Möglichkeiten für uns bereit.

Bild

	<p>Beschreibung</p> <p>Wenn man auf «Columns» klick kommt man auf die Übersicht mit einer Auflistung aller dazugehörige Modelle.</p>
	<p>Beschreibung</p> <p>Wenn man aber auf den Namen der Modelle klickt dann kommt man direkt auf folgende Übersicht. Hier kann man auf «Add Column» Button klicken und Daten erstellen.</p>

So sieht die Erstellung von neuen Daten in Django aus. Dieser Übersicht ist auch konfigurierbar, man kann hier z.B. manche Felder ausblenden.

Tabelle 28 IST-Zustand: Django Admin Übersicht

11.1.8. Fahrplaner Projekt Struktur

Screenshot	Beschreibung
	<p>Obere Übersicht</p> <p>Django verfügt über einen CLI namens django-admin, man kann damit das Projekt erstellen. Es erstellt einen Packet welche standardmäßig gleich wie das Projekt heisst. In Fahrplaner Projekt wurde es jedoch anders benannt, und zwar «core». Zudem wird noch ein manage.py Module erstellt, welches für die Steuerung des Projekts verwendet wird.</p> <p>Dieser Datei verfügt z.B. über die Funktionalität automatisch Daten Migrationen zu kreieren und die Datenbank automatisiert zu erstellen.</p>
	<p>Core Packet</p> <p>Ein Teil davon wurde oben noch erklärt. Dieses Paket dient als Zusammenstellung aller Konfigurationsdateien des Projekts.</p>

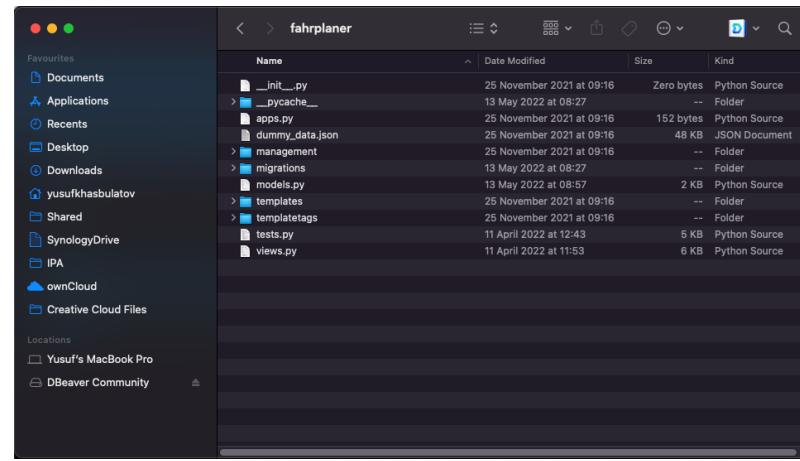
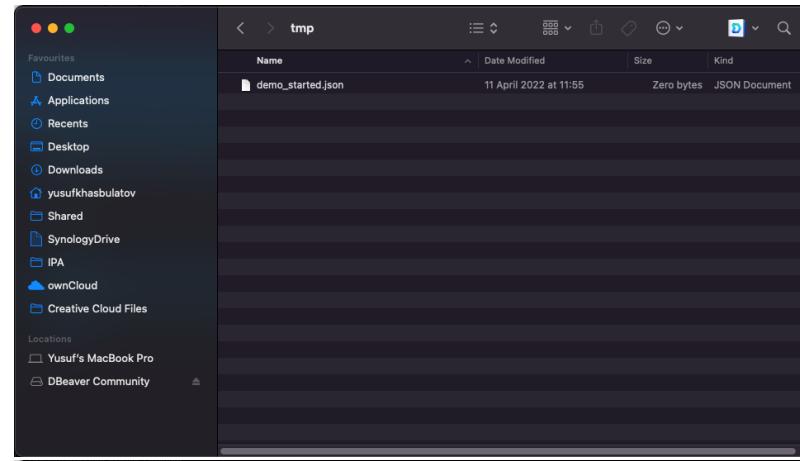
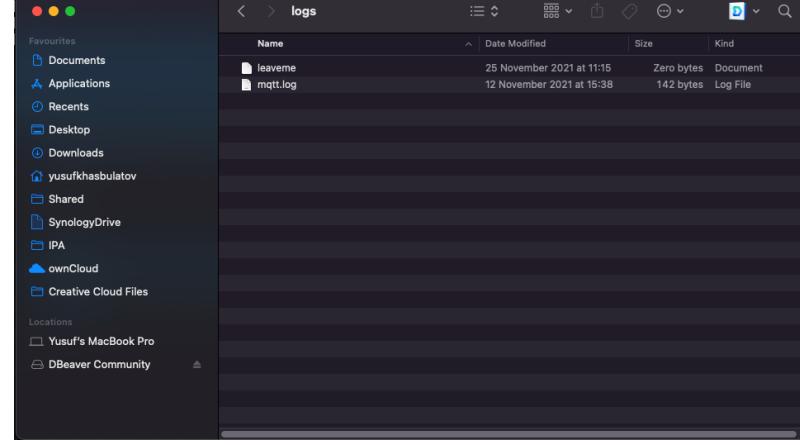
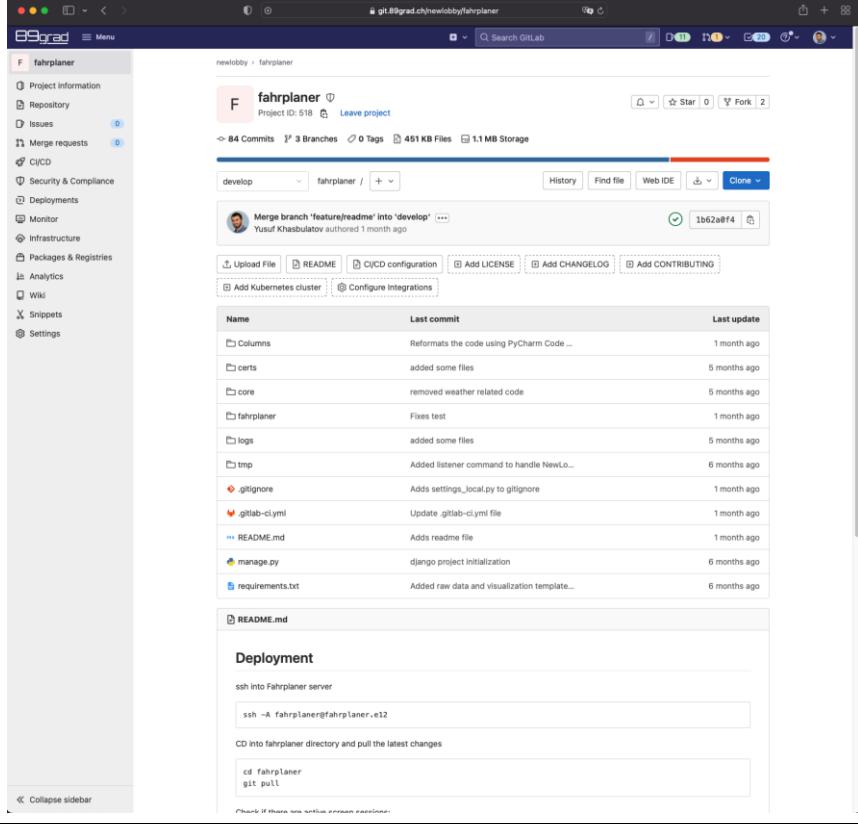
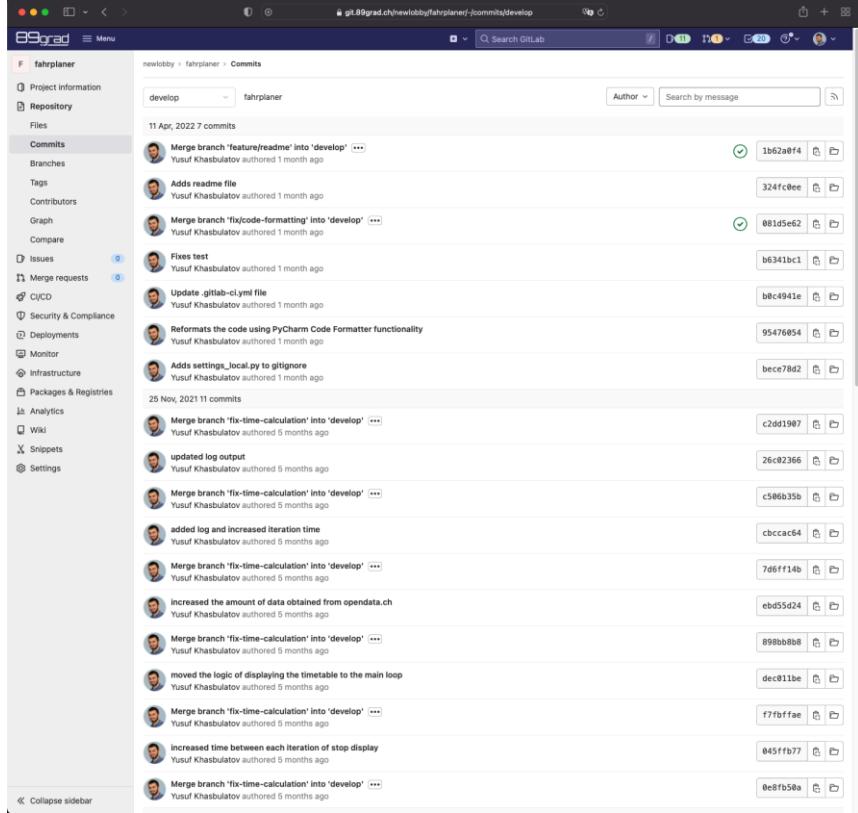
 <table border="1"> <thead> <tr> <th>Name</th><th>Date Modified</th><th>Size</th><th>Kind</th></tr> </thead> <tbody> <tr><td>__init__.py</td><td>25 November 2021 at 09:16</td><td>Zero bytes</td><td>Python Source</td></tr> <tr><td>__pycache__</td><td>13 May 2022 at 08:27</td><td>--</td><td>Folder</td></tr> <tr><td>apps.py</td><td>25 November 2021 at 09:16</td><td>152 bytes</td><td>Python Source</td></tr> <tr><td>dummy_data.json</td><td>25 November 2021 at 09:16</td><td>48 KB</td><td>JSON Document</td></tr> <tr><td>management</td><td>25 November 2021 at 09:16</td><td>--</td><td>Folder</td></tr> <tr><td>migrations</td><td>13 May 2022 at 08:27</td><td>--</td><td>Folder</td></tr> <tr><td>models.py</td><td>13 May 2022 at 08:57</td><td>2 KB</td><td>Python Source</td></tr> <tr><td>templates</td><td>25 November 2021 at 09:16</td><td>--</td><td>Folder</td></tr> <tr><td>templatetags</td><td>25 November 2021 at 09:16</td><td>--</td><td>Folder</td></tr> <tr><td>tests.py</td><td>11 April 2022 at 12:43</td><td>5 KB</td><td>Python Source</td></tr> <tr><td>views.py</td><td>11 April 2022 at 11:53</td><td>6 KB</td><td>Python Source</td></tr> </tbody> </table>	Name	Date Modified	Size	Kind	__init__.py	25 November 2021 at 09:16	Zero bytes	Python Source	__pycache__	13 May 2022 at 08:27	--	Folder	apps.py	25 November 2021 at 09:16	152 bytes	Python Source	dummy_data.json	25 November 2021 at 09:16	48 KB	JSON Document	management	25 November 2021 at 09:16	--	Folder	migrations	13 May 2022 at 08:27	--	Folder	models.py	13 May 2022 at 08:57	2 KB	Python Source	templates	25 November 2021 at 09:16	--	Folder	templatetags	25 November 2021 at 09:16	--	Folder	tests.py	11 April 2022 at 12:43	5 KB	Python Source	views.py	11 April 2022 at 11:53	6 KB	Python Source	<p>fahrplaner Packet (Applikation)</p> <p>Dies ist eine Applikation welche mittels «django-admin» CLI erstellt wurde. Hier ist alle Business Logik der Wackeldisplay Fahrplaner gesammelt.</p>
Name	Date Modified	Size	Kind																																														
__init__.py	25 November 2021 at 09:16	Zero bytes	Python Source																																														
__pycache__	13 May 2022 at 08:27	--	Folder																																														
apps.py	25 November 2021 at 09:16	152 bytes	Python Source																																														
dummy_data.json	25 November 2021 at 09:16	48 KB	JSON Document																																														
management	25 November 2021 at 09:16	--	Folder																																														
migrations	13 May 2022 at 08:27	--	Folder																																														
models.py	13 May 2022 at 08:57	2 KB	Python Source																																														
templates	25 November 2021 at 09:16	--	Folder																																														
templatetags	25 November 2021 at 09:16	--	Folder																																														
tests.py	11 April 2022 at 12:43	5 KB	Python Source																																														
views.py	11 April 2022 at 11:53	6 KB	Python Source																																														
 <table border="1"> <thead> <tr> <th>Name</th><th>Date Modified</th><th>Size</th><th>Kind</th></tr> </thead> <tbody> <tr><td>demo_started.json</td><td>11 April 2022 at 11:55</td><td>Zero bytes</td><td>JSON Document</td></tr> </tbody> </table>	Name	Date Modified	Size	Kind	demo_started.json	11 April 2022 at 11:55	Zero bytes	JSON Document	<p>tmp, log</p> <p>Diese Verzeichnisse dienen als Platzhaltern für die temporäre Dateien und Dateien, die nicht in der Versionierung gespeichert werden sollen.</p>																																								
Name	Date Modified	Size	Kind																																														
demo_started.json	11 April 2022 at 11:55	Zero bytes	JSON Document																																														
 <table border="1"> <thead> <tr> <th>Name</th><th>Date Modified</th><th>Size</th><th>Kind</th></tr> </thead> <tbody> <tr><td>leaveme</td><td>25 November 2021 at 11:15</td><td>Zero bytes</td><td>Document</td></tr> <tr><td>mqtt.log</td><td>12 November 2021 at 15:38</td><td>142 bytes</td><td>Log File</td></tr> </tbody> </table>	Name	Date Modified	Size	Kind	leaveme	25 November 2021 at 11:15	Zero bytes	Document	mqtt.log	12 November 2021 at 15:38	142 bytes	Log File																																					
Name	Date Modified	Size	Kind																																														
leaveme	25 November 2021 at 11:15	Zero bytes	Document																																														
mqtt.log	12 November 2021 at 15:38	142 bytes	Log File																																														

Tabelle 29 IST-Zustand: Projektstruktur

11.1.9. Fahrplaner Testspezifikation

Das bestehende System verfügt über eine Menge an Unit-Tests, welche die Fahrplaner Applikation im Django testen. Es wurden bisher jedoch keine Blackbox Tests gemacht. Für das neue System können die bestehenden Unit-Tests nicht verwendet werden und müssen daher neu geschrieben werden.

11.1.10. Projekt Versionierung

Screenshot	Beschreibung
	<p>Übersicht</p> <p>Das Projekt wurde in 89grad GmbH GitLab Server versioniert. An dem Projekt haben zuvor ich und Herr Baumgartner zusammen gearbeitet.</p>
	<p>Commits</p> <p>Es wurde sich nicht stark an den Standard gehalten, da es sich eher um eine POC handelte und das Ziel darin bestand, die Hardware so schnell wie möglich zum Laufen zu bringen. Daher wurden die Commits nicht an einer Spezifikation vorgenommen.</p>

The screenshot shows the GitLab interface for the 'fahrplaner' repository. The left sidebar contains project information, repository files, commits, branches, tags, contributors, issues, merge requests, CI/CD, security & compliance, deployments, monitor, infrastructure, packages & registries, analytics, wiki, snippets, and settings. The right panel displays the 'Branches' section with three branches: 'develop' (selected), 'minor-adjustments', and 'main'. The 'develop' branch has a merge request and a compare button. The 'main' branch also has a merge request and a compare button.

Branches

Es wurde auch nicht viel mit den Branchen gemacht, Grund dafür war, dass es grundsätzlich nur ein Worker-Skript geplant war welchen das Risiko nach grossen Fehlern sehr klein geschätzt wurde und daher wurde vieles direkt in «development» Branch gepuscht.

Tabelle 30 IST-Zustand: Projekt Versionierung

11.1.11. Einführungsprozess (erste Einführung)

Der Server wurde bereits von einem Mitarbeiter der 89grad GmbH aus der Abteilung Operations konfiguriert und die benötigte Software und Pakete wie python-pip, apache2, git, vim, virtualenv und virtualenvwrapper wurden installiert. In dieser Dokumentation werden die Schritte nicht im Detail beschrieben und daher werden die Pakete, die im Deployment-Prozess verwendet werden, als bereits installierte Pakete betrachtet.

Continuous Integration ist im fahrplaner-Projekt konfiguriert, aber Continuous Deployment ist nicht konfiguriert, daher läuft der Deployment-Prozess manuell.

Screenshot	Schritte/Beschreibung
<pre>\$ ssh -A fahrplan@fahrplan.e12 -i ~/.ssh/id_rsa</pre>	<p>Schritt-1: zuerst muss man sich mit dem fahrplaner Server über SSH verbinden. Dies geschieht mit einem -A-Parameter, um den SSH-Authentifizierungsagenten an den Server weiterzuleiten, damit wir dann das Repository auf dem Server aktualisieren können.</p>

 <pre>\$ git clone git@git.89grad.ch:yusuf.khasbulatov/wd-wetter.git</pre>	Schritt-2: dann muss man die Repository klonen.
 <pre>\$ cd wd-wetter</pre>	Schritt-3: danach muss man sich in das Projektverzeichnis wechseln.
 <pre>\$ mkvirtualenv wetter</pre>	Schritt-4: es ist einen «Best Practice» in Python-Welt immer einen Virtuellen Umgebung zu kreieren, um die Projekt-Pakete zu installieren. Dies wird mittels mkvirtualenv gemacht.
 <pre>\$ workon wetter \$ pip install -r requirements.txt</pre>	Schritt-5: dann muss man die Virtuelle Umgebung aktivieren und mit der Installation der Pakete anfangen.
 <pre>\$ python manage.py migrate</pre>	Schritt-6: danach muss man die Datenbankmigration ausführen. Zu diesem Zweck gibt es die Datei Django manage.py, die diesen Prozess übernimmt.
 <pre>\$ python manage.py initialize</pre>	Schritt-7: nach der Migration erstellen wir die Konfiguration in der Datenbank. Zu diesem Zweck gibt es ein Skript, das vordefinierte Daten in die Datenbank überträgt.
 <pre>\$ touch core/wsgi.py</pre>	Schritt-8: nun kann man das Django Server Starten.
 <pre>\$ screen -S fahrplaner \$ screen -r fahrplaner</pre>	Schritt-9: man muss für den Worker einen Screen Session erstellen und aktivieren.
	Schritt 10: Start des Workers in der Screen-Session.

<pre>\$ python manage.py worker_fahrplaner</pre>	Nachdem man den Worker gestartet hat, kann man den Screen-Session mit der folgenden Tastenkombination freischalten. Ctrl+a dann d
--	---

Tabelle 31 Erste Einführungsschritte

Nun läuft der Worker und man kann den QR-Code scannen, um die Daten auf dem Wackeldisplay anzuzeigen.

11.1.12. Einführungsprozess (Softwareupdate)

Da einige Schritte beim Softwareupdate wie beim Erste Einführungsprozess gehen wird hier auf entsprechende Schritte verwiesen.

Screenshot	Schritte/Beschreibung
 <pre>\$ git fetch \$ git pull</pre>	<p>Folgende Schritte aus Erste Einführungsprozess wiederholen:</p> <p>Schritt-1 dann Schritt-3 und dann Schritt-5</p> <p>Update-Schritt-1: die Repository aktualisieren.</p> <p>Schritt-6 und Schritt-8</p> <p>Danach das Screen Session aktivieren wie im Schritt-9 beschrieben ist.</p> <p>Und der Worker Starten wie im Schritt-10 Beschrieben ist.</p>

Tabelle 32 Softwareupdate Schritte

11.2. Stärken der IST-Situation

Die für den ESP8266 entwickelte Software ist sehr generisch und kann für fast jeden Datensatz verwendet werden. Man muss nur die «Labels» der Symbole in der Datenbank umbenennen und bei Bedarf weitere hinzufügen, und das sollte bei diesem Projekt unverändert so bleiben.

11.3. Schwächen der IST-Situation

Nr.	Kategorie	Beschreibung
SW1	Irrelevante Daten	Die Auswahl des Datenformats, das als Abfahrtszeit des öffentlichen Verkehrs angezeigt werden soll, ist für die aktuelle Version des Displays nicht geeignet. Aufgrund der begrenzten Geschwindigkeit der Datenanzeige kann es vorkommen, dass der Nutzer die nächste Haltestelle des öffentlichen Verkehrs

		nicht mehr rechtzeitig erreichen kann. Oder das betreffende Fahrzeug ist bereits weggefahren.
SW1	Keine Wahlmöglichkeit	Es ist nicht möglich, die anzuzeigenden Daten auszuwählen. Das Display läuft nach dem Start automatisch ab und zeigt alle eingehenden Daten an. Der Benutzer wiederum muss warten, bis das Display die für ihn interessanten Daten anzeigt.
SW3	Keine Anleitung	Im Moment ist nicht klar, was genau getan werden muss, damit das Display startet. Oft stehen die Leute einfach vor dem Display und verstehen nicht, was es tut und was seine Funktionen sind.

Tabelle 33 IST-Zustand: Schwächen

11.4. Persönliche Vorgehensziele

Die persönlichen Vorgehensziele richten sich vor allem nach den Meilensteinen im Zeitplan.

- Ich werde die Vorbereitungen bis am 05.05.2022 fertigstellen.
- Ich werde den Initialisierungs-Teil bis am 06.05.2022 fertigstellen und einen Antrag für die Projekt freigabe machen.
- Ich werde die Konzeptphase bis am 12.05.2022 fertigstellen und einen Antrag für die Phasenfreigabe machen.
- Ich werde die Realisierungsphase bis am 19.05.2022 fertigstellen und einen Antrag für die Phasenfreigabe machen.
- Ich werde bis am 20.05.2022 eine Benutzeranleitung erstellen.
- Ich werde das Projekt testen und einen Antrag für die Projektabnahme machen.

11.5. Projektziele

Nr.	Ziel	Abdeckt
Z1	Es muss eine geeignete, kostenlose und dokumentierte Datenquelle bestimmt werden.	SW1
Z2	Die Daten müssen im JSON Format sein.	SW1
Z3	Die Daten müssen für einen bestimmten Ort sein.	SW1, SW2
Z4	Die Daten müssen die Temperatur für einen bestimmten Ort zeigen können.	SW1, SW2
Z5	Die Daten müssen den Stand des Wetters für einen bestimmten Ort zeigen können.	SW1, SW2
Z6	Die Daten müssen den Niederschlag in mm/h für bestimmte Ort zeigen können.	SW1, SW2
Z7	Die Daten müssen eine Tendenz für die kommenden 3 Tage beinhalten.	SW1, SW2
Z8	Die Daten müssen in einer Tabellenform auf der Webseite dargestellt werden.	SW2
Z9	Ausgewählte Daten müssen auf dem Wackeldisplay anzeigen werden können.	SW2

Z10	Es muss eine Benutzeranleitung erstellt werden, welche aufzeigt wie das Display gestartet werden kann.	SW3
Z11	Es muss eine Benutzeranleitung erstellt werden, welche aufzeigt wie das Display die Daten für einen bestimmten Ort zeigen kann.	SW3

Tabelle 34 Projektziele

SW1...SW3 – steht in der Tabelle oben für «Schwäche»

11.6. Grenzwerte

Aus der [Detaillierte Aufgabenstellung](#) und nach dem Abgleich im [Daily Meeting](#) sind folgende Grenzwerte klar:

Orte: Bern, Zürich, Paris, London, San Francisco

Trend: Der Trend soll aus den Werten der nächsten drei Tage berechnet werden. Grundsätzlich soll der Trend in Form eines Pfeils dargestellt werden, der je nach Situation nach oben, nach unten oder gerade nach rechts zeigt.

Die verbleibenden Grenzwerte sind noch zu klären und werden in der Konzeptphase detailliert betrachtet.

11.7. Abgrenzungen

Die Software, die auf dem esp8266 installiert ist, wird von mir weder entwickelt noch verbessert. Es handelt sich um eine in C++ geschriebene Software, mit der ich keine Erfahrung habe.

11.8. Anforderungen

Die Anforderungen beschreiben die Rahmenbedingungen für das Projekt. Die Informationen müssen aus der detaillierten Projektbeschreibung entnommen werden.

11.8.1. Funktionale Anforderungen

Nr.	Beschreibung	Akzeptanzkriterium	Abdeckt
F1	Die ausgewählte Datenquelle sendet die Responses im JSON Format zurück	Die Daten können mittels python.json() in ein JSON Objekt umgewandelt werden.	Z2
F2	JSON String beinhaltet einen Ort	Es muss möglich sein, entweder die Daten für folgende Städte zu filtern: Bern, Zürich, Paris, London, San Francisco; oder eine direkte Anfrage für die erwähnten Städte zu machen.	Z3
F3	JSON String beinhaltet die Information über die Temperatur für einen bestimmten Ort.	Es muss möglich sein, aus dem JSON Objekt die Temperatur für einen ausgewählten Ort herauszukriegen	Z4
F4	JSON String beinhaltet die Information über den Wetterzustand eines bestimmten Ortes.	Das JSON muss unterschiedliche Wetterzustand-Daten beinhalten, damit verschiedene Symbole auf dem	Z5

		Wackeldisplay angezeigen werden können.	
F5	JSON String beinhaltet die Information über den Niederschlag.	Es muss eine Information über den Niederschlag beinhalten, welche man in Einheit mm/h umwandeln kann.	Z6
F6	Die ausgewählte Datenquelle sendet die Daten für mehrere Tage.	Es muss ein Datenformat sein, welches in eine python list umgewandelt werden kann.	Z7

Tabelle 35 Funktionale Anforderungen

11.8.2. Nicht funktionale Anforderungen

Nr.	Beschreibung	Akzeptanzkriterium	Abdeckt
NF1	Die Datenquelle ist gut dokumentiert.	Die Information für die richtigen Anfragen und Formate muss vorhanden sein.	Z1
NF2	Die Daten sind für die Darstellung auf einer Webseite vorbereitet.	Die Daten müssen für die Darstellung in einer HTML Tabelle geeignet sein.	Z8
NF3	Es besteht ein Steuerungselement auf der Webseite.	Jede Zeile der Tabelle hat einen Button, welcher die Daten in dieser Zeile an das Wackeldisplay sendet.	Z9
NF4	Es besteht eine Broschüre oder Flyer mit der Anleitung für das Wackeldisplay	Es muss eine A5-Benutzeranleitung oder ein A4-Flyer geschrieben werden, welche am Wackeldisplay beigelegt oder angeklebt werden können.	Z10, Z11

Tabelle 36 Nicht Funktionale Anforderungen

11.9. Risikoanalyse

Eine allgemeine Risikoanalyse welche technischen und sozialen Aspekte berücksichtigt. Die Risiken sind von den technischen zu den sozialen Risiken geordnet, aber nicht kategorisiert.

ID	Risikobeschreibung	Auswirkung	Vor Massnahme				Massnahmen/Erklärung	Nach Massnahme			
			W	S	Risiko	Handlungsweise		W	S	Risiko	Handlungsweise
R1	Datenquelle Verbindungsprobleme.	Zeitverlust	W2	S4	Mittel	Risikominderung	Arbeit an der Dokumentation, wenn möglich.	W2	S1	Niedrig	Risikoakzeptanz
R2	Die Anzahl von möglichen API-Anfragen laufen ab.	Zeitverlust	W3	S3	Hoch	Risikominderung	Weiteres Benutzerkonto erstellen. Es muss jedoch eine Datenquelle sein, welche keine Kreditkarten Information verlangt.	W3	S1	Niedrig	Risikoakzeptanz
R3	Lokaler Netzwerkunterbruch.	Zeitverlust	W1	S4	Mittel	Risikominderung	Arbeit an der Dokumentation, wenn möglich.	W1	S1	Niedrig	Risikoakzeptanz
R4	Abbruch der Internetverbindung	Zeitverlust	W1	S3	Mittel	Risikominderung	Eine Internetverbindung via Mobiltelefon herstellen, bis das Problem behoben ist.	W1	S1	Niedrig	Risikoakzeptanz
R5	Probleme mit dem ESP8266-Mikrocontroller.	Zeitverlust	W3	S2	Mittel	Risikominderung	Für jede Spalte Ersatz-Microcontroller vorbereiten. Heisst, Software installieren, MAC-Adresse notieren.	W3	S1	Niedrig	Risikoakzeptanz
R6	Arbeitslaptop funktioniert nicht mehr	Zeitverlust	W2	S4	Mittel	Risikominderung	Einen Ersatzlaptop vorbereiten. Während der Vorbereitung noch an der Dokumentation arbeiten.	W2	S1	Niedrig	Risikoakzeptanz
R7	Krankheit/Unfall	Zeitverlust	W1	S4	Mittel	Risikominderung	Sofort dem Hauptexperten melden und ein Zeugnis vom Arzt besorgen. Weiteres Vorgehen mit dem Hauptexperten besprechen.	W1	S1	Niedrig	Risikoakzeptanz
R8	Fehlplanung von Aufgaben und daraus resultierende Über- oder Unterschätzung.	Zeitverlust	W2	S3	Mittel	Risikominderung	Dieses Risiko lässt sich nie ganz vermeiden, aber es kann gemanagt werden. Aufgaben verschieben, Reservezeit einplanen.	W2	S1	Niedrig	Risikoakzeptanz

Tabelle 37 Projekt Risikoanalyse

Legende: R1...R8-Risikoid S-Schadensausmass; W-Wahrscheinlichkeit

Schadensausmass

Abkürzung	Beschreibung
S1	führt zu keiner Abwertung
S2	geringe Abwertung (bis 1.0 Notenpunkte)
S3	hohe Abwertung (über 1.0 Notenpunkte)
S4	führt zu nicht bestehen

Tabelle 38 Legende: Schadensausmass

Eintrittswahrscheinlichkeit

Abkürzung	Beschreibung
W1	Unwahrscheinlich (<20%)
W2	Eventuell (20-49%)
W3	Wahrscheinlich (50-80%)
W4	Sehr wahrscheinlich (>80%)

Tabelle 39 Legende: Eintrittswahrscheinlichkeit

Risikomatrix

Risikomatrix als Hilfsmittel welche alle mögliche Zahlen nach Schadensausmass und Eintrittswahrscheinlichkeit zusammen Rechnung darstellt. Hier gilt: Risiko = Schadensausmass × Eintrittswahrscheinlichkeit

4	4	8	12	16
3	3	6	9	12
2	2	4	6	8
1	1	2	3	4
1	2	3	4	

Tabelle 40 Risikomatrix

Risikokategorisierung

Die möglichen Zahlen werden aufgeteilt nach Kategorie

1	2	3	4	6	8	9	12	16
Niedrig			Mittel			Hoch		

Tabelle 41 Risikomatrix Beschriftungen

11.10. Risikograph

		<20%	20-49%	50-80%	>80%	
		R3, R7	R1, R6			führt zu Nicht-bestehen
		R4	R8	R2		Abwertung über 1.0 Notenpunkte
				R5		Abwertung bis 1.0 Notenpunkte
						führt zu keiner Abwertung

		<20%	20-49%	50-80%	>80%	
		R3, R4 R7	R1, R6, R8	R2, R5		führt zu Nichtbestehen
						Abwertung über 1.0 Notenpunkte
						Abwertung bis 1.0 Notenpunkte
						führt zu keiner Abwertung

Tabelle 42 Risikograph

Legende: R – steht für Risikonummer

11.10.1. Kurze Stellungnahmen zu den Risiken

Die von mir identifizierten Risiken überschreiten eine Wahrscheinlichkeit von 3 nicht. Aber die Risiken selbst sind sehr gefährlich und können, wenn sie stillschweigend hingenommen werden, zum Nichtbestehen der Prüfung führen. Ich habe über mögliche Lösungen für dieses oder jenes Problem nachgedacht und entsprechende Massnahmen zur Verringerung der Gefahr des Nichtbestehens der Prüfung vorgesehen.

11.11. Lösungsvarianten

Hier wird festgelegt, wie das Projekt anschliessend angegangen werden soll. Es werden 4 Varianten betrachtet und die Entscheidung wird auf Basis der erfüllten Kriterien getroffen.

11.11.1. Variante 1 - Tomorrow.io

Die API von Tomorrow.io bietet einen All-in-One-Endpunkt mit 60 verschiedenen Datenfeldern, einschliesslich Wetter, Luftqualität, Pollen, Strassenverkehrsrisiko und Feuerindex, und umfasst ausserdem historische, Echtzeit- und Vorhersagedaten zum Wetter auf der ganzen Welt. Jedoch sind nicht alle Funktionen kostenlos.

11.11.2. Variante 2 - OpenWeatherMap

Die OpenWeatherMap API bietet derzeit eine Vielzahl von Wetterdaten, einschliesslich (aber nicht beschränkt auf) aktuelles Wetter, Vorhersagen, historische Daten, Wetterstationen und Wetterwarnungen. Dieser Quelle wird auch von der Community sehr beliebt. Es ermöglicht 1000000 Anfragen pro Monat und 60 pro Tag.

11.11.3. Variante 3 - Weatherstack

Weatherstack - wird laut ihrer Webseite von ziemlich grossen Unternehmen wie Microsoft, Schneider Electric und Warner Bros. genutzt. Ihre API bietet auch einen Wetterverlauf bis zum aktuellen Tag. Auch die Geschwindigkeit und die Genauigkeit sind den Bewertungen auf verschiedenen Webseiten zufolge gut. Ein grosser Nachteil ist, dass sie die Einrichtung eines Kontos mit Bankkartendaten erfordern.

11.11.4. Variante 4 - Meteotest

Die Meteotest-Wetter-API liefert die neuesten Wetter-, Klima- und Umweltdaten im Ausgabeformat: CSV, XML, JSON oder YAML. Es ist auch möglich, die Ausgabedaten anzupassen.

Durch die Vorverarbeitung der benötigten Daten wird eine hohe Geschwindigkeit und Verfügbarkeit auch bei grossen Datenabfragen erreicht. Diese Ressource hat den Vorteil, dass es sich um eine Schweizer Ressource handelt, aber leider gibt es keinen freien Zugang.

Variantenvergleich

Kriterien	Variante 1	Variante 2	Variante 3	Variante 4
Hat eine Dokumentation	1	2	2	1
Kostenlos	1	2	0	0
Mit Authentifizierung (API Key)	2	2	2	2
Benötigt persönliche Daten (Kreditkarte)	2	2	0	0
Geeignetes Datum Format	1	1	2	2
Genügend Anfragen möglich?	1	2	0	0
Passt zu Django (Python)	2	2	1	1
Kommt aus der Schweiz	0	0	0	2
Total	10	13	7	8

Tabelle 43 Variantenvergleich API-Quellen

Bewertung	Beschreibung
0	Nein, Kriterium nicht erfüllt
1	Kriterium teilweise erfüllt
2	Ja, Kriterium erfüllt

Tabelle 44 Bewertungsschema

11.12. Variantenantrag mit Begründung

Aus dem Vergleich geht OpenWeatherMap klar als klarer Favorit hervor. Bis auf zwei Einschränkungen werden alle Kriterien erfüllt. So bietet OpenWeatherMap die Daten in einem unpassenden Datumsformat an. Dieses Format kann allerdings mit wenig Aufwand in Python umgewandelt werden. Der zweite Nachteil ist, dass OpenWeatherMap kein Schweizer Anbieter ist. Allerdings sind über OpenWeatherMap auch Schweizer Daten verfügbar, so dass dieser Nachteil nicht so stark ins Gewicht fällt. Der einzige Schweizer Anbieter (Variante 4) bietet leider keinen kostenlosen Zugang an und scheidet deswegen aus.

12. Konzept

In der Konzeptentwicklung werden die Grundlagen für die Realisierung und Einführung eines IT-Systems erarbeitet.

Die bestehende Wackeldisplay-Software wird durch die Erstellung einer neuen Anwendung (Paket) in Django erweitert, die auf Wetterdaten ausgerichtet ist. Der Code der bestehenden Software wird ebenfalls verbessert (wo nötig Kommentare einfügen, Code nach Coding-Convention überprüfen).

Benutzerschnittstelle und Designkonzept

Das Wackeldisplay selbst ist die Schnittstelle, über die der Benutzer die Software steuern kann. Die bestehenden Symbole werden durch neue Symbole ersetzt. Das Design wird dem Farbschema der 89grad GmbH folgen. Es wird eine Webseite entwickelt, auf der die Daten in tabellarischer Form dargestellt werden. Für die Webseite werden zudem Interaktive Elemente entwickelt, damit der User ausgewählte Daten auf dem Wackeldisplay anzeigen lassen kann. Diese Webseite wird ebenfalls dem Farbschema der 89grad GmbH folgen.

Systemarchitektur

Dieses Projekt besteht aus einer Kombination von Hardware und Software: Wackeldisplay und Django-Anwendung. Die ESP8266-Microcontrollers dienen als Schnittstelle zwischen den beiden Seiten, die Befehle über das WLAN lesen und an den Server senden.

Qualitätssicherung

Alle Funktionen und Anwendungsfälle werden mit Tests überprüft. Positive Tests werden durchgeführt, um erwartete Rückgabewerte und Verhaltensweisen zu prüfen, und negative Tests werden durchgeführt, um unerwartete Rückgabewerte und Verhaltensweisen zu prüfen.

12.1. Systemanforderungen

Wackeldisplay

Das Wackeldisplay wurde bereits in der [Studie; IST-Zustand](#) Kapitel beschrieben: Hinzu kommen neu nur die Symbole.

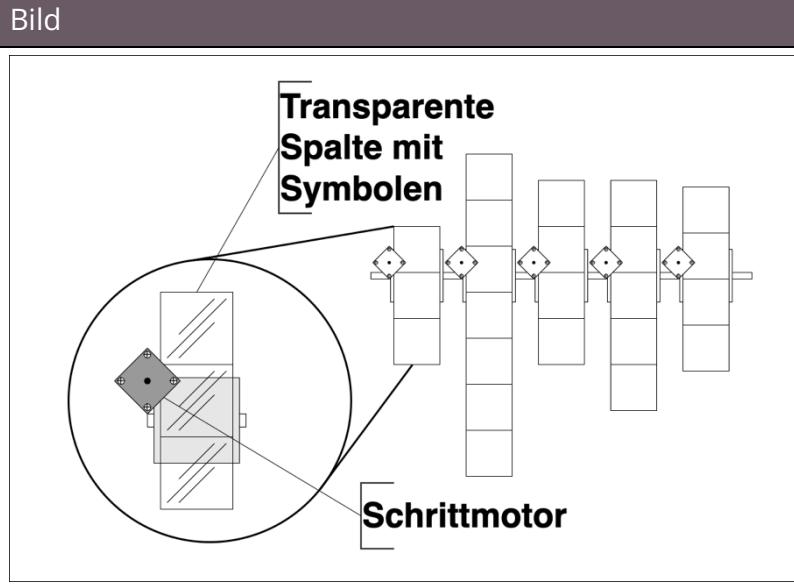
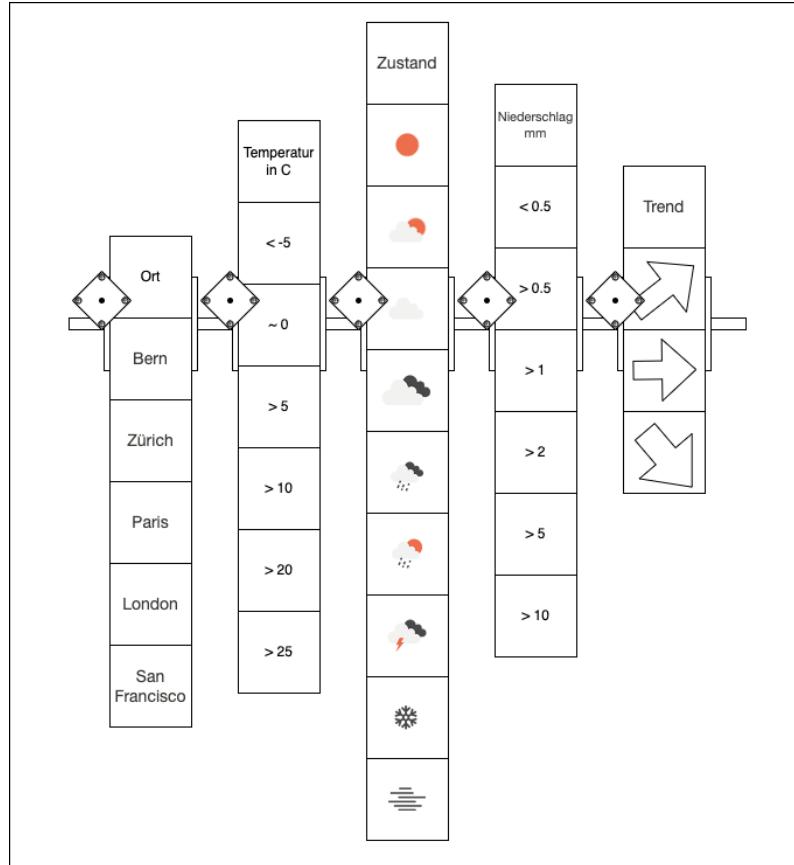
Bild	Beschreibung
	Bestehendes Wackeldisplay mit 5 Spalten
	Das Konzept für die Symbole. Die Bilder wurden von der OpenWeatherApi übernommen und ins 89grad GmbH Farbschema überführt.

Tabelle 45 Konzept: Wackeldisplay Symbole

12.1.1. Grenzwerten Konzept

Ort

Orte werden gemäss [Detaillierte Aufgabenstellung](#) erstellt.

Temperatur

Siehe Sitzungsprotokoll 4

Zustand

Siehe Sitzungsprotokoll 4

Niederschlag

Um die entsprechenden Niederschlagswerte herauszufinden, wurden die täglichen Niederschlagsmittelwerte der letzten 12 Monate von der Website von MeteoSchweiz analysiert und die folgenden Balkendiagramme erstellt:

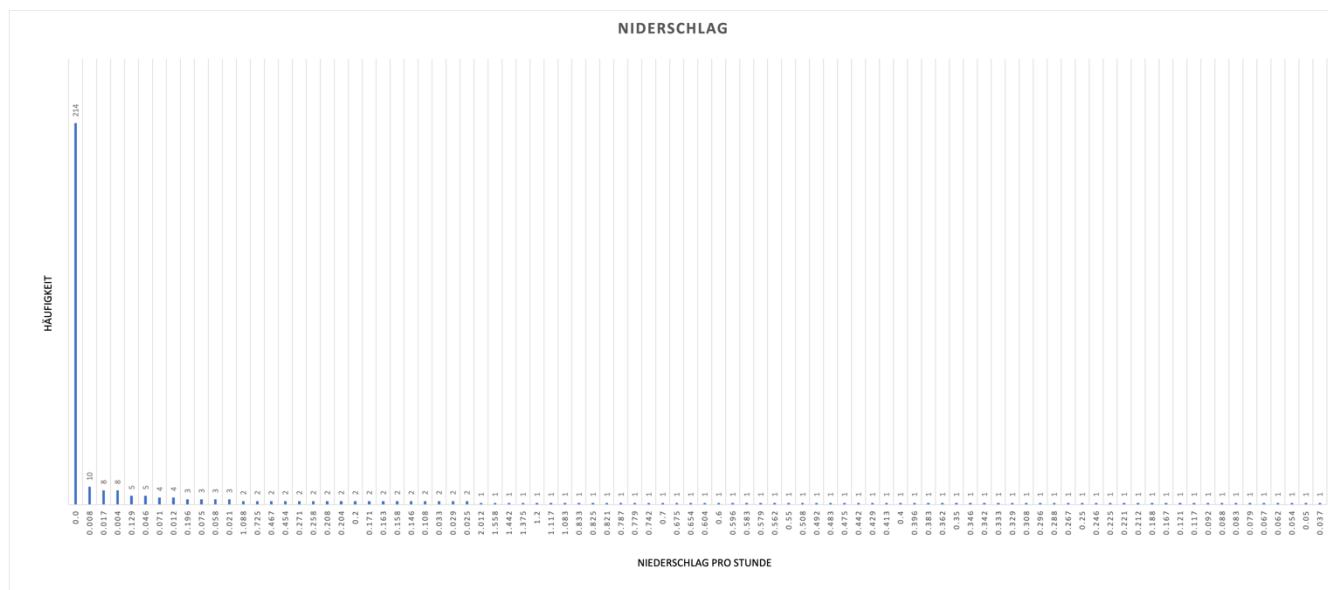


Abbildung 26 Niederschlag pro Tag geteilt durch Stunde der letzten 12 Monate

Die obige Abbildung zeigt die durchschnittliche Niederschlagsmenge der letzten 12 Monate in Form eines Balkendiagramms. Die Daten stammen, wie bereits erwähnt, von der Website von MeteoSchweiz.

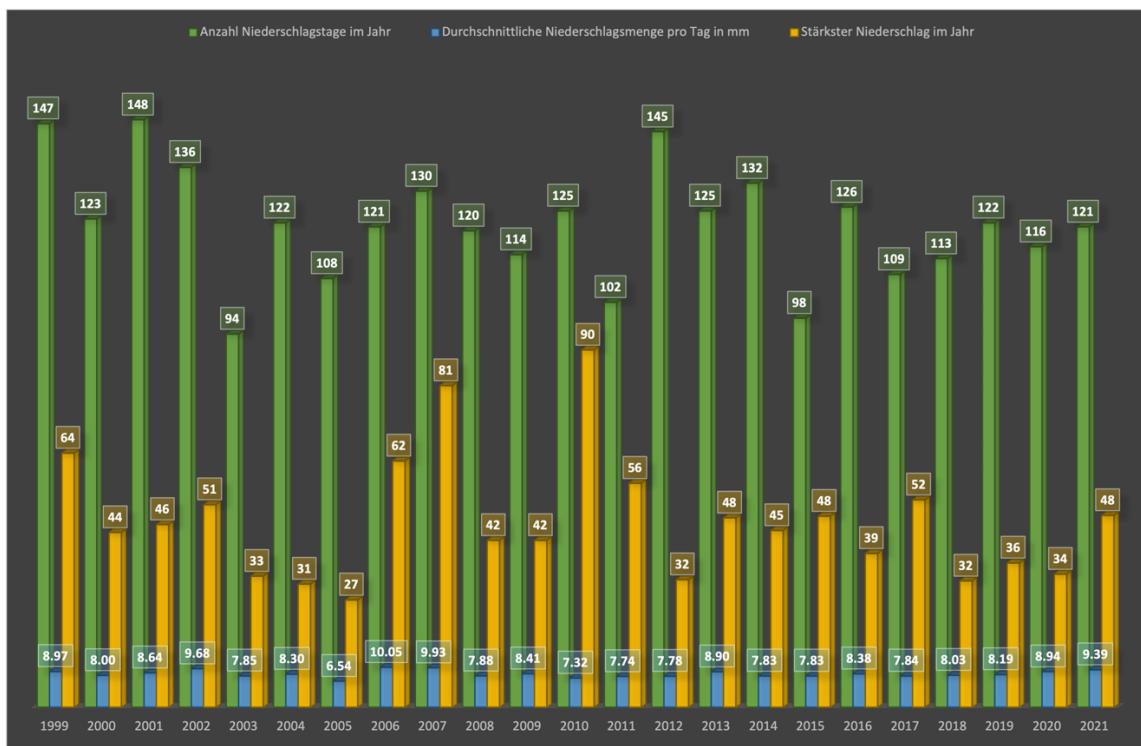


Abbildung 27 Niederschlag: Historische Daten in 10 Jahren

Die obige Abbildung zeigt die durchschnittliche Niederschlagsmenge der letzten 10 Jahre. Dieses Balkendiagramm zeigt, wie viele Niederschlagstage es in bestimmten Jahren gab, in welchem Jahr es am meisten Niederschlag gab und wie hoch die durchschnittliche Niederschlagsmenge pro Tag in bestimmten Jahren war.

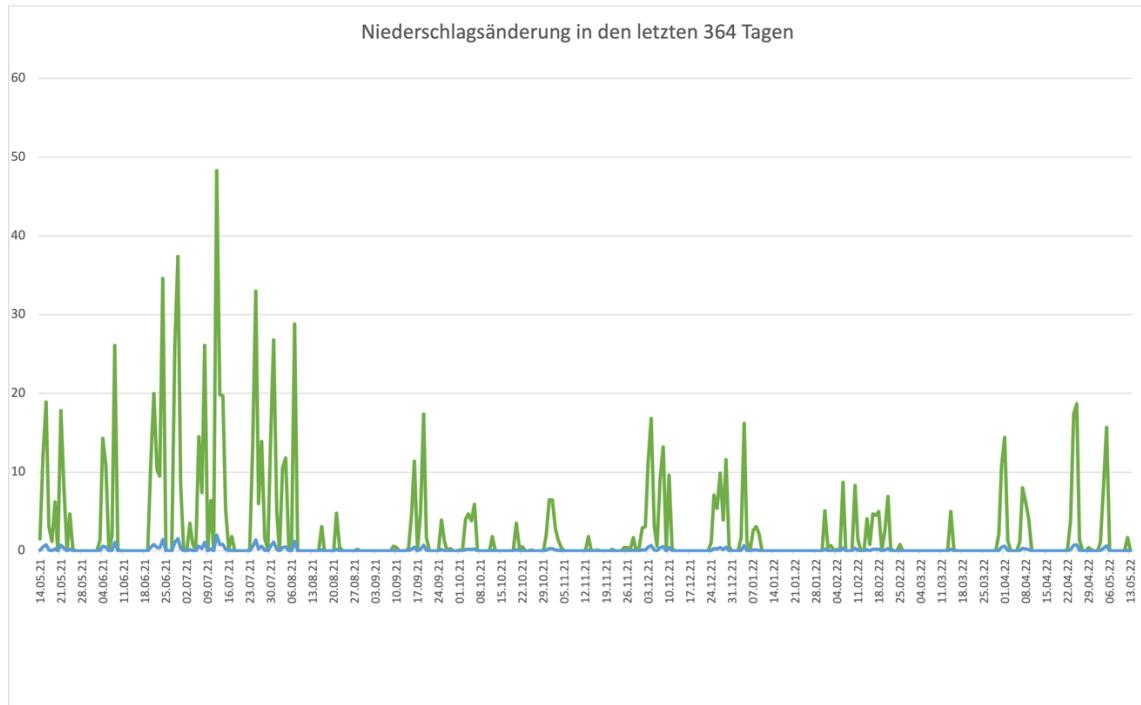


Abbildung 28 Niederschlagsänderung in Letzte 12 Monaten

Das obige Liniendiagramm zeigt die Veränderung der durchschnittlichen täglichen Niederschlagsmenge in mm während der letzten 12 Monate.

Trend

Jedes Symbol erhält ein Token in Form einer Zahl, wobei Tokens als ein Wert zu betrachten sind. Auf der Spalte werden neun Symbole aufgeklebt, daher gibt es Wertmarken von 1 bis 9. Hier ist eine Liste aller Token:

Symbol Name	Symbol Wert	Icon
clear sky	9	
few clouds	8	
mist	7	
scattered clouds	6	
broken clouds	5	
rain	4	
shower rain	3	
thunderstorm	2	
snow	1	

Tabelle 46 Konzept: Grenzwerte - Beschreibung der Wetter Symbole

Die Berechnung erfolgt in folgendem Szenario: Heute ist es sonnig (clear sky), also Token 9, morgen wird es regnen (shower rain) Token 3, übermorgen wird es leicht regnen (rain) Token 4 und überübermorgen wird es bewölkt (scattered clouds) Token 6. Nach der Berechnung des Durchschnitts der nächsten drei Tage erhalten wir 4.3:

$$(3+4+6) / 3 = 4.3$$

Da 4.3 kleiner als 9 ist (heute ist 9), zeigen wir einen Pfeil nach unten.

$$\left(\text{cloud} + \text{cloud} + \text{cloud} \right) / 3 = 4.3$$

Tabelle 47 Konzept: Formel für die Trendberechnung

12.1.2. Anwendungsfälle

Die bestehenden Anwendungsfälle des IST-Zustands unterscheiden sich gegenüber dem neuen System nicht wesentlich. Es wird neue Funktion eingebaut, um bestimmte Daten anzuzeigen. Daher gibt es neu die Funktion «Bestimmte Daten anzeigen» welche in unteren Use-Case Diagramm zu sehen ist.

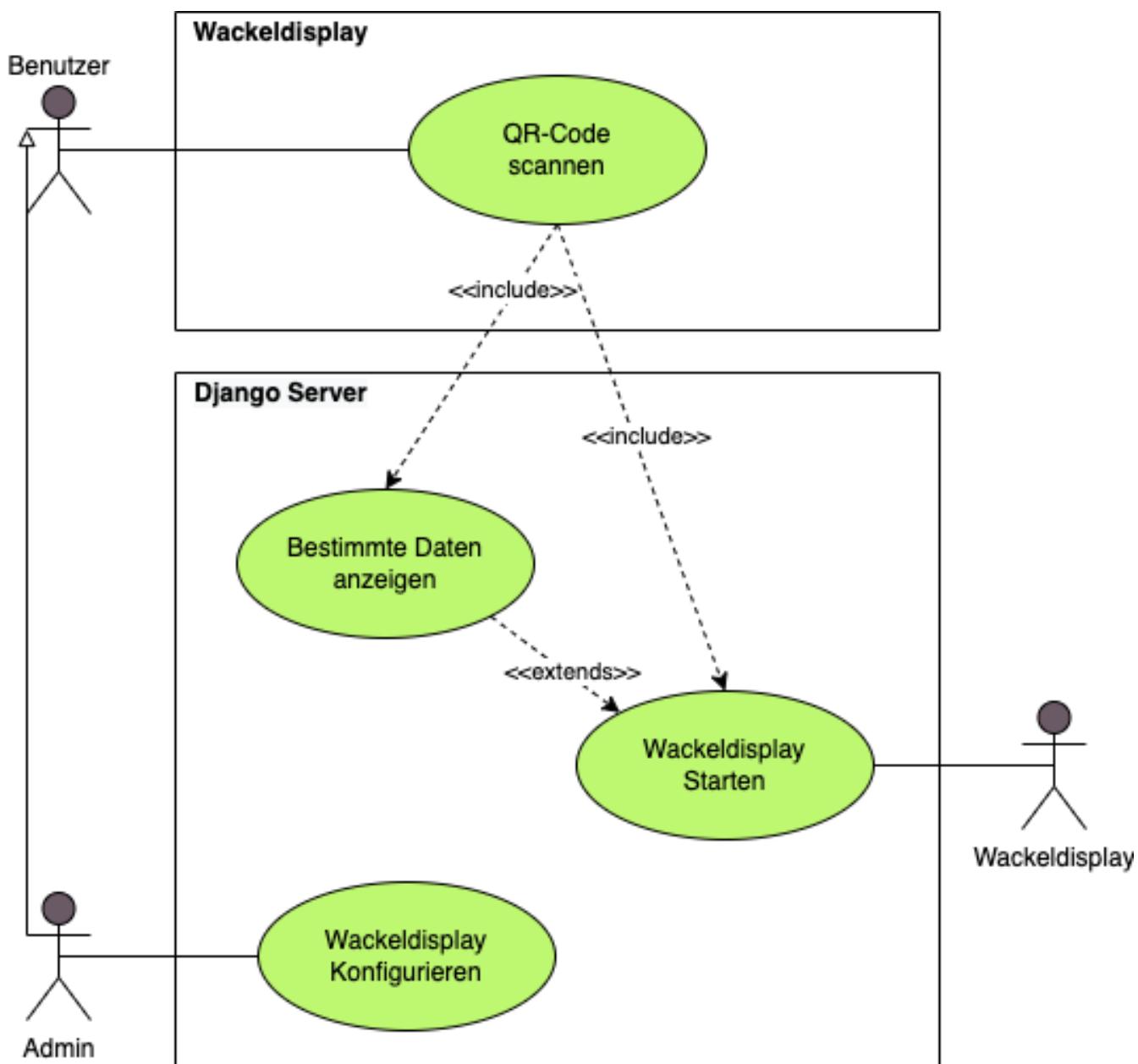


Abbildung 29 Konzept: Anwendungsfälle

Die Beschreibung von «QR-Code scannen», «Wackeldisplay Konfigurieren» und «Wackeldisplay starten» sind in dem Kapitel [Studie; IST-Zustand](#) zu finden.

Anwendungsfall «Wetterdaten auf der Webseite zeigen»	
Kurzbeschreibung	Der Anwender/in öffnet die Webseite, indem er/sie auf den Link klickt, welcher nach dem QR-Code Scan erscheint.
Akteure	Jeder
Vorbedingungen	<ul style="list-style-type: none"> - Einen QR-Code muss vorhanden sein. - Das Wackeldisplay muss eingeschaltet sein. - Der Anwender/in muss mit dem 89grad GmbH WiFi verbunden sein. - Der Wackeldisplay Service muss auf dem Server gestartet sein. - Der Django Server muss gestartet sein. - Die Wackeldisplay Konfiguration muss vorhanden sein. - Alle Bedingungen auf der Server Seite müssen erfüllt sein.
Ablauf	<ul style="list-style-type: none"> - Der Anwender/in scannt den QR-Code mit einem Handy. - Der Anwender/in öffnet die URL.
Resultat	Die Daten werden auf der Webseite in einer Tabelle angezeigt.
Ausnahmen	Keine Ausnahmen.
Anwendungsfall «Bestimmte Daten anzeigen»	
Kurzbeschreibung	Der Anwender/in öffnet die Webseite, indem er/sie auf den Link klickt, welche nach dem QR-Code Scan erscheint. Dann kann der/die Anwender/in sich bestimmte Daten an der Wackeldisplay anzeigen lassen, in dem er auf den entsprechenden Button klicken.
Akteure	Jeder
Vorbedingungen	<ul style="list-style-type: none"> - Einen QR-Code muss vorhanden sein. - Das Wackeldisplay muss eingeschaltet sein. - Der Anwender/in muss mit dem 89grad GmbH WiFi verbunden sein. - Der Wackeldisplay Service muss auf dem Server gestartet sein. - Der Django Server muss gestartet sein. - Die Wackeldisplay Konfiguration muss vorhanden sein. - Alle Bedingungen auf der Server Seite müssen erfüllt sein.
Ablauf	<ul style="list-style-type: none"> - Der Anwender/in scannt den QR-Code mit einem Handy. - Der Anwender/in öffnet die URL. - Der Anwender/in klickt auf den Button in Tabellenzeile.
Resultat	Die ausgewählten Daten werden auf dem Wackeldisplay angezeigt.
Ausnahmen	Wenn das Wackeldisplay bereits in Betrieb ist, wird diese Aktion nicht durchgeführt.

Tabelle 48 Konzept: Wackeldisplay Anwendungsfall

12.1.3. Datenbankkonzept

Das Django Web Framework erstellt standardmäßig die Datenbank, die es braucht, um richtig zu funktionieren. Nachfolgend ein ausgewählter Ausschnitt des ERD (in modifizierter Chen-Notation), welches Django automatisch erstellt (die Namen der Entitäten sind vereinfacht).

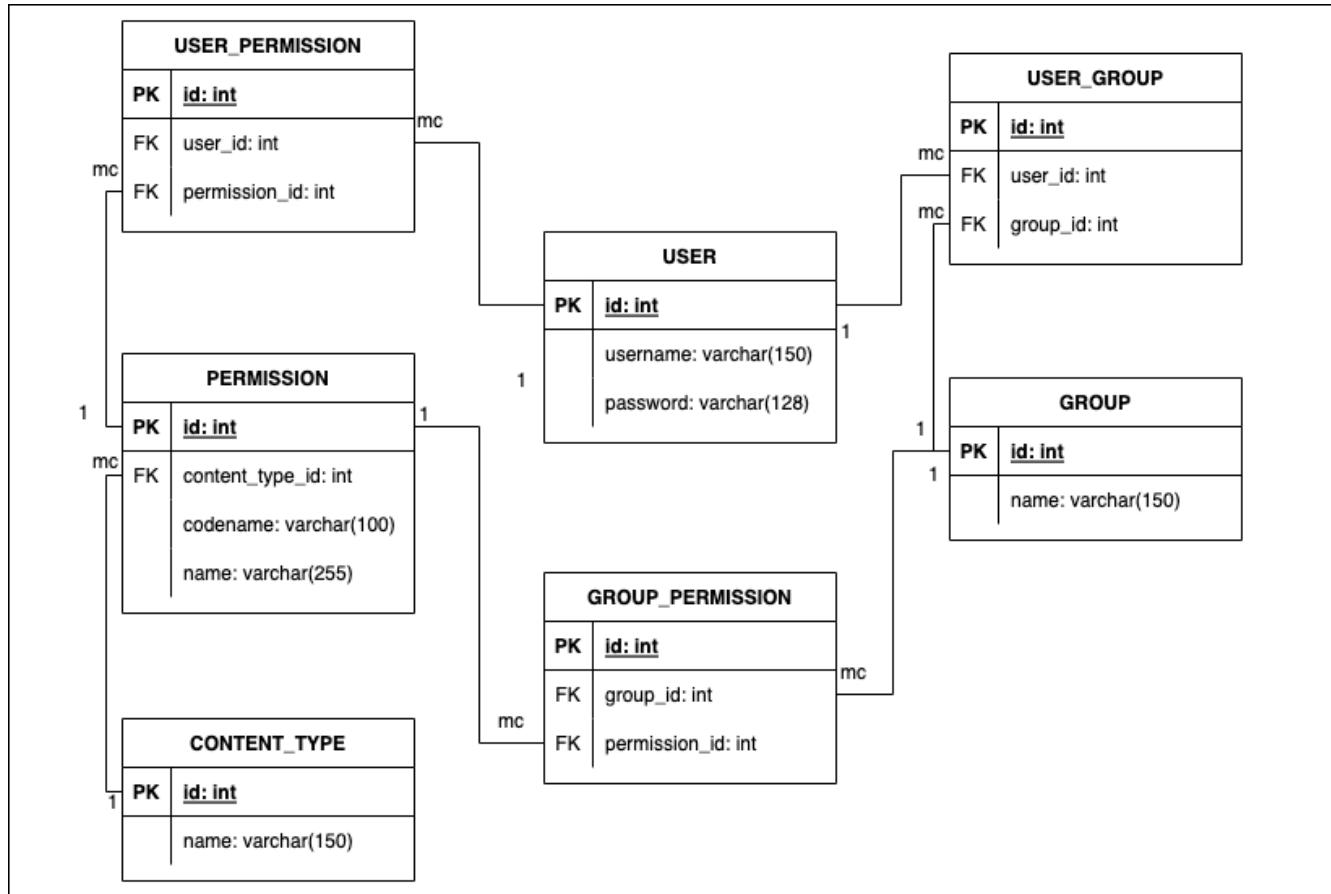


Abbildung 30 Konzept: Django ERD

Tabelle	Beschreibung
User_Permission, User_Group, Group_Permission	Dies sind Zwischentabellen für die M2M-Beziehung.
User	Hier werden alle Benutzerdaten gespeichert.
Permission	Hier werden standartmäßig verschiedene Berechtigungen von Django erstellt. Zudem kann der Admin hier seine eigenen Berechtigungen definieren.
Group	Dieser Tabelle dient für die Erstellung der Groups. Derselbe Benutzer kann zu mehreren Groups hinzugefügt werden. Die Berechtigungen können daher sehr flexibel verwaltet werden.
Content_Type	Hier wird ein Pfad zu jeder Komponente/jedem Modell als Zeichenkette gespeichert. Die Berechtigungen besagen, ob der Benutzer Zugriff auf diesen oder jenen Pfad hat.

Tabelle 49 Konzept: Django Datenbank Beschreibung

Weiterhin sind bereits zwei zusätzliche Entitätstypen vorhanden, welche von Herrn Florian Baumgartner erstellt wurden (siehe [Datenbank \(SQLite\)](#)).

Um sicherzustellen, dass das Wackeldisplay auch funktioniert, wenn z.B. die öffentliche Datenquelle offline geht oder aus irgendeinem anderen Grund nicht mehr funktioniert, werden alle Daten in der Datenbank gespeichert. Für die Abspeicherung der Daten wird folgendes Datenmodell erstellt.

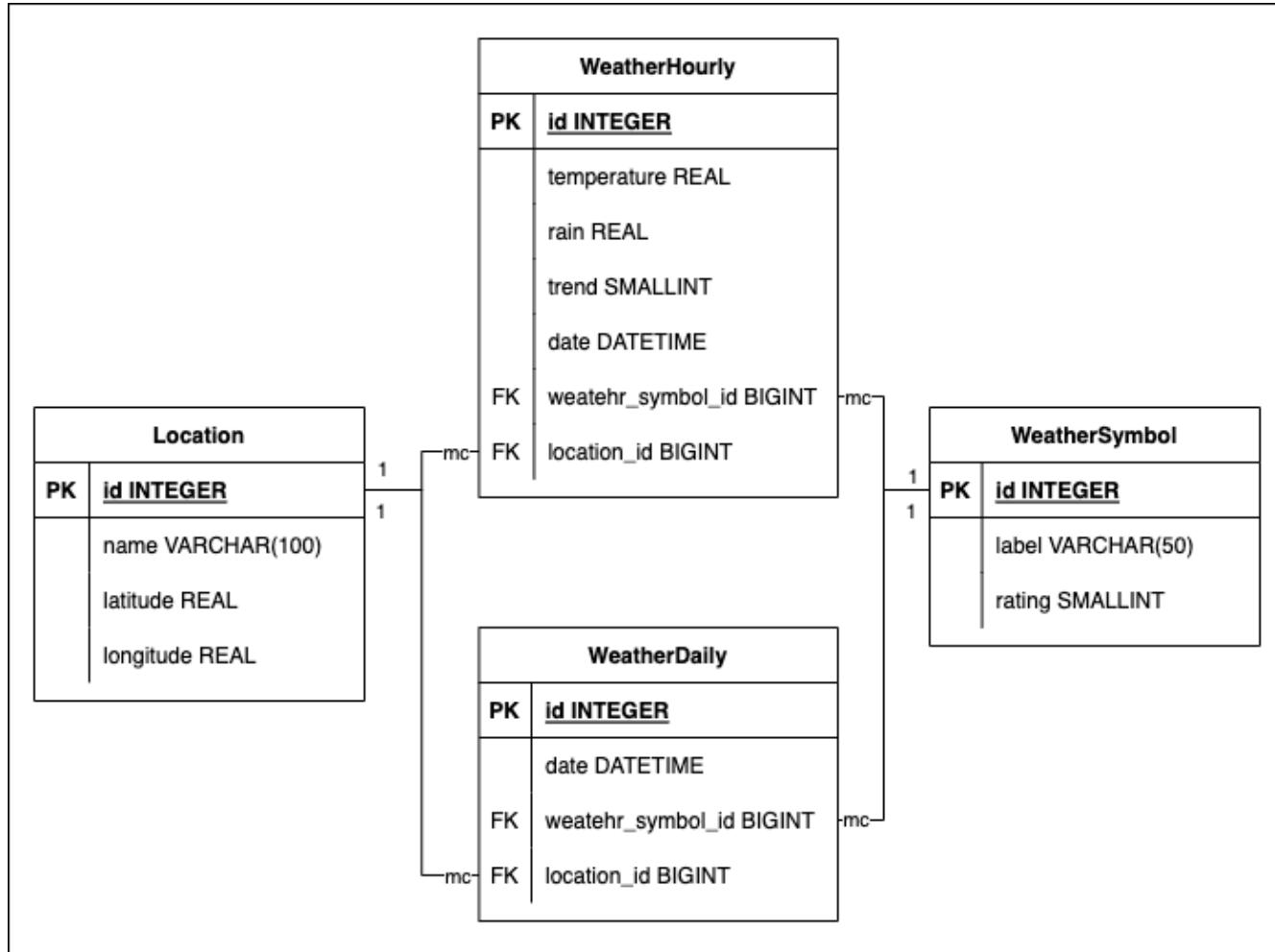


Abbildung 31 Konzept: Wetter ERD

Tabelle	Beschreibung
Location	Alle Standorte werden hier gespeichert. Wie in der Aufgabenstellung angegeben, sind derzeit nur 5 Standorte geplant. Die Tatsache, dass später weitere Standorte hinzugefügt werden können sollen, setzt eine dynamische Lösung dieser Aufgabe voraus. Ich werde dies bei der Erstellung von Diensten für die Arbeit mit dem Wackeldisplay-API berücksichtigen.
WeatherHourly	Hier werden die Daten gespeichert, die stündlich vom Dienst angefordert werden. Das Wackeldisplay wiederum bezieht seine Daten nicht mehr aus der API-Ressource selbst sondern aus dieser Tabelle.
WeatherDaily	Hier werden die Daten für die nächsten 3 Tage gespeichert. Der Dienst wird nicht jedes Mal alle drei Tage abfragen. Sondern, nachdem die ersten

	drei Tage ausgefüllt wurden, nur n+1. Das bedeutet, dass er nur einen Tag hinzufügen muss, und zwar jeden Tag. Danach werden diese Daten für die Berechnung des Trends verwendet.
WeatherSymbol	WeatherHourly und WeatherDaily benötigen beide Symbole. Um Redundanz zu vermeiden, wird diese Tabelle eingeführt. Hier werden die Symbole gespeichert: Nicht direkt die Symbole als z.B. base64-String, sondern nur der Name und die Bewertung. Die Symbole können als statische Daten im Projekt gespeichert werden.

Tabelle 50 Konzept: Wetter ERD Beschreibung

12.2. Benutzerschnittstelle und Designkonzept

Auf der «Home Page» wird die Tabelle dargestellt und der Benutzer kann auf den «Zeigen» Button klicken, um sich die Daten auf dem Wackeldisplay anzeigen zu lassen.

Ort	Temperatur in C	Zustand	Niederschlag in mm	Aktion
Bern	14	Sonne	0	Zeigen
Zürich	13	Sonne	0	Zeigen
London	7	Wolken	0	Zeigen
Paris	8	Regnet	0.20	Zeigen
San Francisco	18	Sonne	0	Zeigen

Abbildung 32 Das "Home Page" Designkonzept

Wenn man auf den «Zeigen»-Button klickt, wird ein «Alert» angezeigt, auf dem die «Zeigen»-Aktion bestätigen werden muss. Das «Alert» ist als Zwischenschritt nötig, damit überprüft werden kann, ob das Wackeldisplay bereits in Betrieb ist und um entsprechend ein Feedback geben zu können.

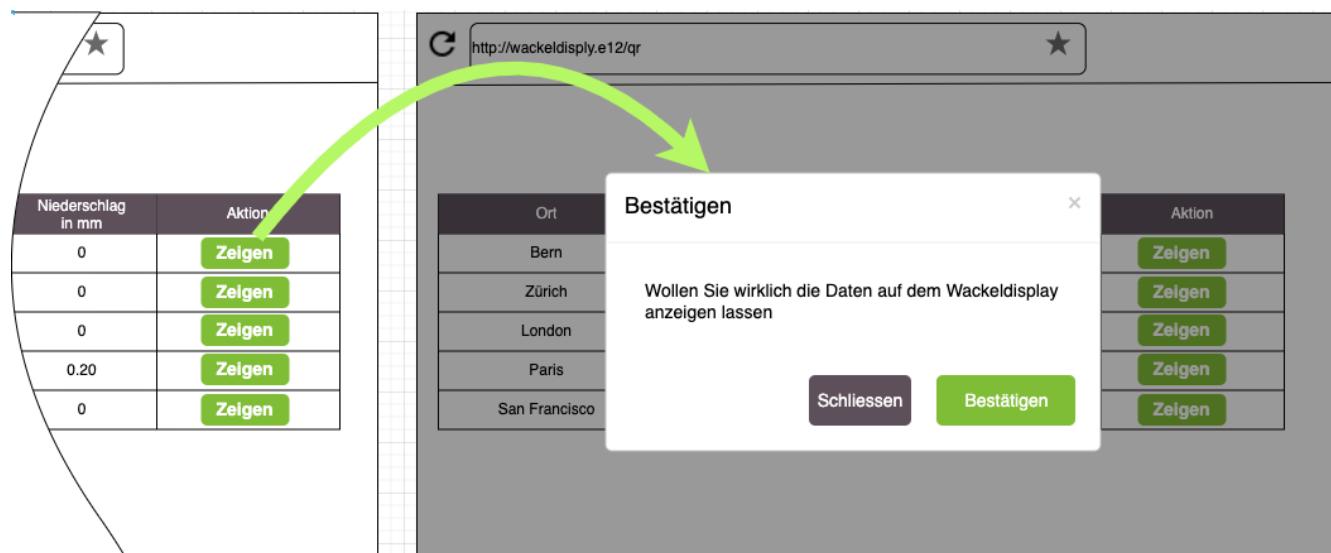


Abbildung 33 Konzept: Wireframe Daten zeigen

Wenn das Wackeldisplay bereits in Betrieb ist, wird das «Alert»-Fenster angezeigt.

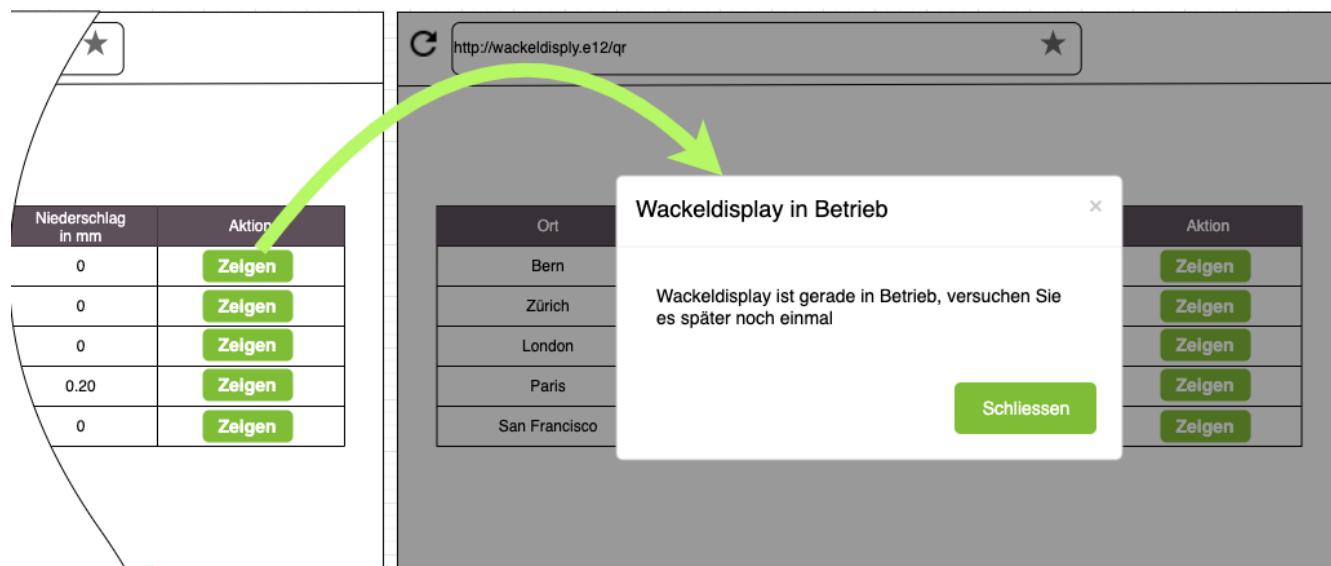


Abbildung 34 Konzept: Wireframe Daten zeigen. Bestätigen

12.2.1. Farbkonzept

Alle Schnittstellen (Webseite, Wackeldisplay Symbole) werden gemäss den [89grad GmbH Design Guidelines](#) gestaltet sein. Ausnahmen gelten nur für Icons, die von der OpenWeatherAPI-Ressource geladen werden.

12.2.2. Schriftkonzept

In diesem Projekt wird die Schriftart Open Sans verwendet, welche den 89grad GmbH Standards entspricht (siehe [Schriftart](#)).

12.3. Materialbeschaffung

Die neuen Symbole und Spalten werden über das FabLab (Tochterfirma der 89grad GmbH) bezogen. Es werden folgende Symbole hergestellt:

Symbole	Fälle, in denen das Symbol angezeigt wird:
	Klar <ul style="list-style-type: none"> • klarer Himmel
	Wolken/Sonne <ul style="list-style-type: none"> • geringe Bewölkung: 11-25%
	Wolken <ul style="list-style-type: none"> • aufgelockerte Bewölkung: 25-50%
	Dichte Wolken <ul style="list-style-type: none"> • durchbrochene Bewölkung: 51-84% • bedeckte Bewölkung: 85-100%
	Regen <ul style="list-style-type: none"> • Regenschauer • Regenschauer mit hoher Intensität • zerklüftete Regenschauer Nieselregen <ul style="list-style-type: none"> • Sprühregen Regen • Regenschauer Nieselregen
	Regen <ul style="list-style-type: none"> • leichter Regen
	Gewitter <ul style="list-style-type: none"> • Gewitter mit leichtem Regen • leichtes Gewitter • Gewitter mit Nieselregen • Gewitter mit starkem Nieselregen
	Schnee <ul style="list-style-type: none"> • leichter Schnee • Starker Schnee • Schneeregen
	<ul style="list-style-type: none"> • Nebel • Rauch

Tabelle 51 Konzept: Symbolen Beschreibung

12.4. Systemarchitektur

Die Schnittstellen, die im [IST-Zustand](#) beschrieben wurden, werden nicht geändert. Die bestehende Situation und die Systemarchitektur gelten auch für die neue Funktionalität. Neu wird eine Django Applikation⁷ erstellt namens «wetter», um die Business Logik zusammenzufassen. Das alte «fahrplaner» Paket (Applikation) wird gelöscht und Columns auf columns umbenannt. Nach den Änderungen wird die Architektur so aussehen:

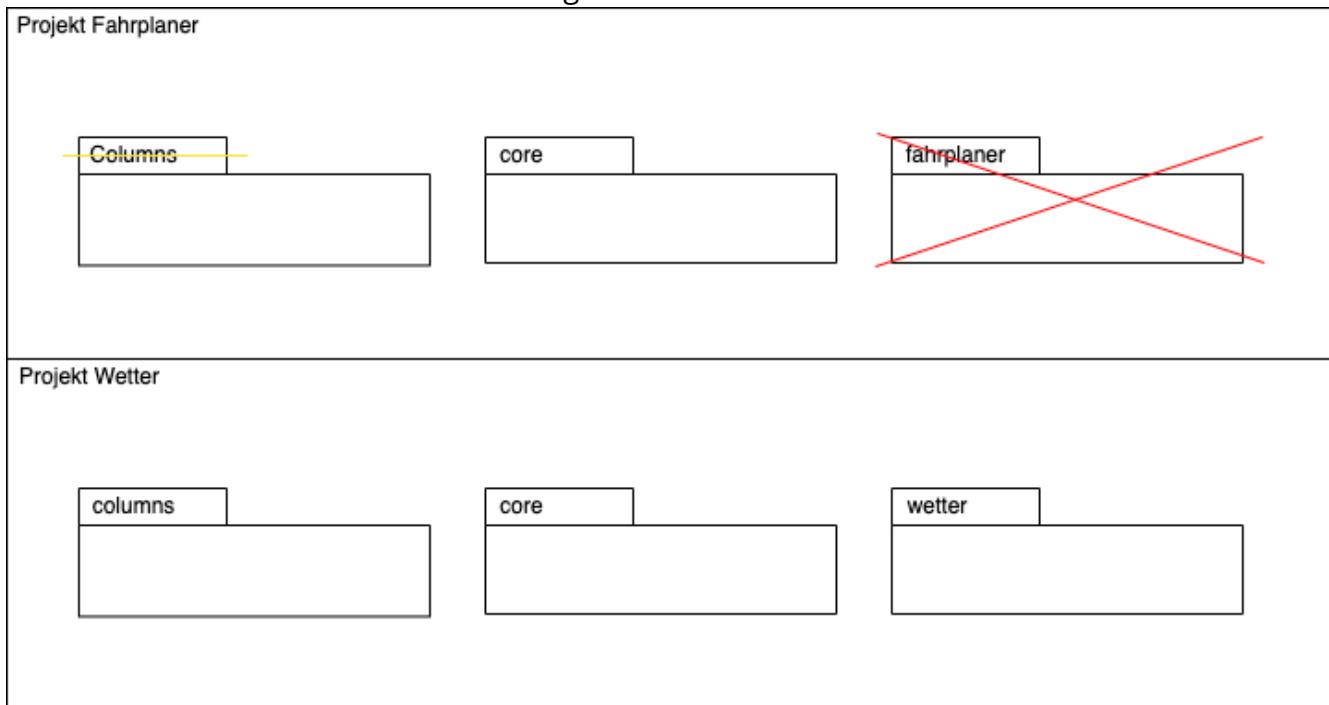


Abbildung 35 Konzept: Django Projekt Architektur vor und nach

12.5. Testkonzept

Um die Funktionalität und Qualität sicher zu stellen, werden White-Box und Black-Box Tests durchgeführt. Getestet wird die [Wackeldisplay Funktionalität](#) und die Funktionalität, welche die [Websei_Benutzerschnittstelle_und_Designkontakte](#) zur Verfügung stellt.

12.5.1. Testmethoden

Die Tests bestehen aus zwei Teilen: Unit-Tests mit einem White-Box-Verfahren und Funktionstests mit einem Black-Box-Verfahren. Funktionstests erfordern Benutzereingaben und visuelle Bestätigung. Unit-Tests erfordern Interaktion im Code und die Überprüfung von Rückgabewerten.

12.5.2. Testumfeld und Testmittel

Die Tests werden in den Räumlichkeiten der 89grad GmbH durchgeführt, da das System nur über das interne Netz funktioniert. Nach erfolgreichen White-Box-Tests wird die Anwendung bereitgestellt und die Black-Box-Tests werden durchgeführt. Als White-Box-Tests werden die Unit-Tests direkt [auf dem PC des Entwicklers](#) durchgeführt. Die Daten für die Unit-Tests werden hartkodiert gespeichert, um die Anzahl der API-Anfragen pro Tag zu reduzieren. Es werden die gleichen Daten wie in der API-Quelle verwendet. Sobald diese Tests erfolgreich abgeschlossen

⁷ Django nennt die Pakete als «Applikation», man erstellt die Applikationen, um bestimmte Business Logik zusammen zu fassen.

sind, wird die Anwendung in der Umgebung einer virtuellen Maschine mittels GitLab Continuous Integration getestet, wobei die virtuelle Maschine mit derjenigen identisch ist, auf der die Anwendung zu einem späteren Zeitpunkt installiert sein wird.

Die Blackbox-Tests werden von einem Mitarbeiter der 89grad GmbH durchgeführt, der mit dem System selbst nicht vertraut ist, aber das Wackeldisplay bereits vor der Installation der neuen Funktionalität genutzt hat. Als Testgerät wird ein Smartphone mit QR-Code-Erkennung verwendet.

12.5.3. Testkonzept Abgrenzungen

Die Software, die für den ESP8266-Mikrocontroller entwickelt wurde, wird nicht direkt von mir getestet. Der Grund dafür ist, dass es sich um eine völlig andere Entwicklungsumgebung handelt. Sie verwendet eine andere Entwicklungssoftware und erfordert zusätzlichen Aufwand, um den Programmiercode zu verstehen. Darüber hinaus ist es in der Computersprache C++ geschrieben, mit der ich keine Erfahrung habe.

12.5.4. Testfälle

Funktionstests

ID/Bezeichnung	FT 1 QR-Code wird als einen Link erkannt.
Beschreibung	Der QR-Code, welcher auf dem Wackeldisplay zu finden ist, soll ein Handy nach dem Scannen als Link erkennen.
Abgedeckte Anwendungsfall	Anwendungsfall «QR-Code scannen»
Ausgangssituation	Der Tester/in steht vor dem Wackeldisplay und ist bereit den QR-Code mit einem Handy zu scannen.
Vorbereitungsschritte	<ol style="list-style-type: none"> Der Tester/in muss sich mit dem Wi-Fi namens «89grad» oder «89guest» verbinden.
Testschritte	<ol style="list-style-type: none"> Der Tester/in scannt den QR-Code auf dem Wackeldisplay.
Erwartetes Ergebnis	Der QR-Code wird als Link erkannt.

Tabelle 52 Testfälle: FT 1 QR-Code wird als einen Link erkannt

ID/Bezeichnung	FT 2 Wackeldisplay startet.
Beschreibung	Nach dem Scan des QR-Codes und Öffnen der URL startet das Wackeldisplay.
Abgedeckte Anwendungsfall	Anwendungsfall «Wackeldisplay Starten»
Ausgangssituation	Der Tester/in steht vor dem Wackeldisplay. Das Wackeldisplay ist auf der initialen Position.
Vorbereitungsschritte	<ol style="list-style-type: none"> Der Tester/in muss sich mit dem Wi-Fi namens «89grad» oder «89guest» verbinden. Der Tester/in muss den QR-Code scannen.

Testschritte	<ol style="list-style-type: none"> Der Tester/in öffnet die URL (auf manchen Handy-Versionen kann es sein, dass die URL automatisch geöffnet wird).
Erwartetes Ergebnis	Das Wackeldisplay startet innerhalb von 1-10 Sekunden.

Tabelle 53 Testfälle: FT 2 Wackeldisplay startet

ID/Bezeichnung	FT 3 Daten werden auf der Webseite angezeigt.
Beschreibung	Nach dem Scan des QR-Codes und Öffnen der URL erscheint die Webseite mit der Darstellung der Daten in Tabellenform.
Abgedeckte Anwendungsfall	Anwendungsfall «Wetter Daten auf der Webseite zeigen»
Ausgangssituation	Der Tester/in steht vor dem Wackeldisplay. Es spielt keine Rolle, ob das Wackeldisplay bereits in Betrieb ist oder nicht.
Vorbereitungsschritte	<ol style="list-style-type: none"> Der Tester/in muss sich mit dem Wi-Fi namens «89grad» oder «89guest» verbinden. Der Tester/in muss den QR-Code scannen.
Testschritte	<ol style="list-style-type: none"> Der Tester/in öffnet die URL (auf manchen Handy-Versionen kann es sein, dass die URL automatisch geöffnet wird). Der Tester/in stellt sicher, dass die Daten auf der Webseite angezeigt werden.
Erwartetes Ergebnis	Die Daten werden auf der Webseite in eine Tabelle dargestellt.

Tabelle 54 Testfälle: FT 3 Daten werden auf der Webseite angezeigt

ID/Bezeichnung	FT 4 Bestimmte Daten werden auf dem Wackeldisplay angezeigt.
Beschreibung	Wenn man auf den Button «Zeigen» in einer bestimmten Zeile klickt, werden die ausgewählten Daten auf dem Wackeldisplay angezeigt.
Abgedeckte Anwendungsfall	Anwendungsfall «Bestimmte Daten anzeigen»
Ausgangssituation	Der Tester/in hat die Webseite geöffnet und sieht die Tabelle mit Daten. Das Wackeldisplay ist auf der initialen Position.
Vorbereitungsschritte	<ol style="list-style-type: none"> Der Tester/in muss sich mit dem Wi-Fi namens «89grad» oder «89guest» verbinden. Der Tester/in muss den QR-Code scannen. Der Tester/in muss die Webseite öffnen.
Testschritte	<ol style="list-style-type: none"> Der Tester/in klickt auf den Button «Zeigen» in der gewünschten Zeile. Der Tester/in klickt auf den Button «Bestätigen», der sich in dem erscheinenden Modal befindet.
Erwartetes Ergebnis	Das Wackeldisplay startet innerhalb von 1-10 Sekunden.

Tabelle 55 Testfälle: FT 4 Bestimmte Daten werden auf der Wackeldisplay angezeigt

ID/Bezeichnung	FT 5 Bestimmte Daten werden auf der Wackeldisplay nicht angezeigt.
Beschreibung	Wenn man auf den Button «Zeigen» einer bestimmten Zeile klickt, werden die ausgewählten Daten auf dem Wackeldisplay nicht angezeigt, weil es bereits in Betrieb ist.
Abgedeckte Anwendungsfall	Anwendungsfall «Bestimmte Daten anzeigen»
Ausgangssituation	Der Tester/in hat die Webseite geöffnet und sieht die Tabelle mit den Daten. Das Wackeldisplay ist in Betrieb.
Vorbereitungsschritte	<ol style="list-style-type: none">1. Der Tester/in muss sich mit dem Wi-Fi namens «89grad» oder «89guest» verbinden.2. Der Tester/in muss den QR-Code scannen.3. Der Tester/in muss die Webseite öffnen
Testschritte	<ol style="list-style-type: none">1. Der Tester/in klickt auf den Button «Zeigen» in der ausgewählten Zeile.2. Der Tester/in klickt auf den Button «Bestätigen», der sich in dem erscheinenden Modal befindet.
Erwartetes Ergebnis	Das Wackeldisplay wird nicht gestartet.

Tabelle 56 Testfälle: FT 5 Bestimmte Daten werden auf der Wackeldisplay nicht angezeigt

13. Realisierung

Zur Erklärung der Anwendung werden hier das Sequenzdiagramm und der Ablauf der Entwicklung dargestellt.

13.1. Wackeldisplay Hardwareanpassung

Das Wackeldisplay wurde ebenfalls für die neuen Datensätze angepasst. Die Symbole sind lediglich vorübergehend befestigt worden und werden vor der Systempräsentation durch die finalen Symbole ersetzt.



Tabelle 57 Realisierung: Wackeldisplay

13.2. Projekt Struktur

Screenshot	Beschreibung
	Obere Übersicht Es hat sich nicht vieles geändert im Vergleich zum IST-Zustand. Neu gibt es die Ordner wetter Applikation, Templates Verzeichnis und Static Verzeichnis .
	Core Packet In diesem Packet hat sich nichts geändert. Es hat sich jedoch inhaltlich verändert. Siehe Anhang settings_devel.py und urls.py
	wetter Packet (Applikation) Dies ist eine Applikation, welche mittels «django-admin» CLI erstellt wurde. Hier ist alle Business Logik des Wackeldisplay Wetter gesammelt. Die Dateien sind im Anhang zu finden.
	tmp, log In diesem Verzeichnis hat sich mit Ausnahme eines Punktes nichts geändert: Der Worker pull_hourly.py legt im logs Verzeichnis eine Datei für die Übertragung der Daten an.

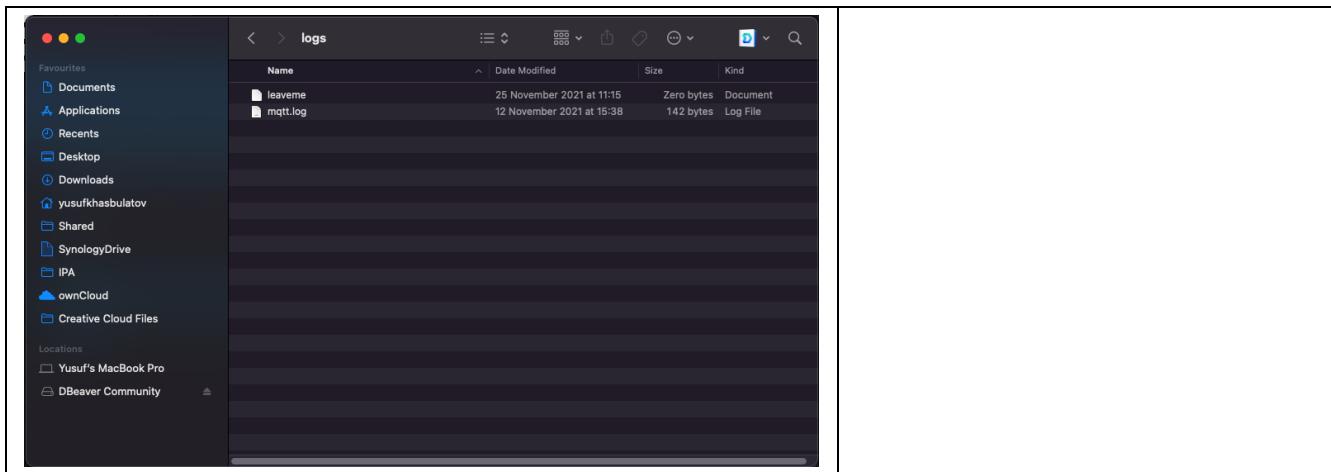


Tabelle 58 Realisierung: Projekt Struktur

13.3. Automatisation und Migration Skripte

Für die schnelle Einführung wurden mehrere Python Skripte geschrieben, welche die Daten-Erstellung und initiale Konfiguration erleichtern.

Nachfolgend ist das Sequenzdiagramm des initialize.py-Skripts dargestellt. Es ist jedoch vereinfacht, um Platz zu sparen:

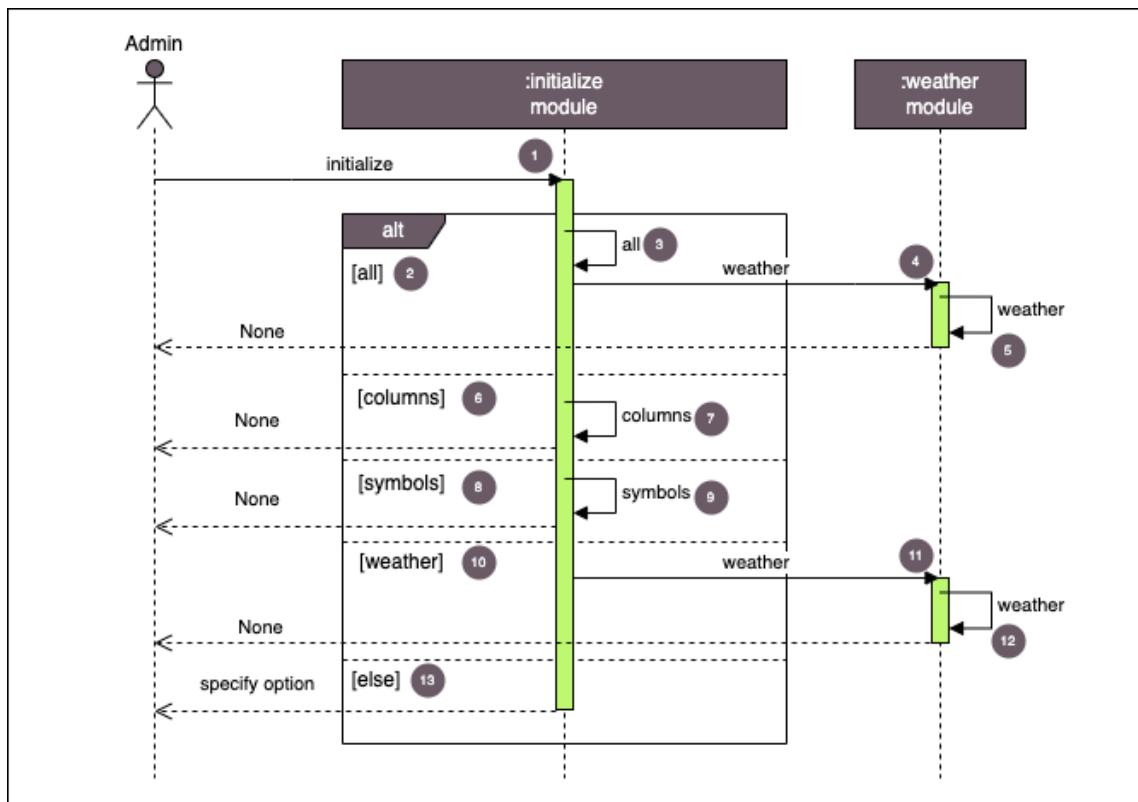


Abbildung 36 Realisierung: Initialize Skripte Sequenzdiagramm

Nr.	Beschreibung
1	Admin führt den Befehl initialize aus.
2	Es wird geprüft, ob der Befehl mit der Option --all ausgeführt wurde.
3	Falls der Befehl mit der Option --all ausgeführt wurde, werden die Spalten- und Symboldateien im Modul Initialize erstellt.

4	Danach wird programmatisch ein weiterer Befehl ausgeführt, der die Wetter-Anwendung konfiguriert und die benötigten Daten erstellt.
5	Als Rückgabewert wird nur die Konsolenausgabe angegeben.
6	Es wird geprüft, ob der Befehl mit der Option --columns ausgeführt wurde.
7	Falls der Befehl Initialize mit der Option --columns ausgeführt wurde, werden die Daten für Columns erstellt und in der Konsolenausgabe angegeben.
8	Es wird geprüft, ob der Befehl mit der Option --symbols ausgeführt wurde.
9	Falls der Befehl Initialize mit der Option --symbols ausgeführt wurde, werden die Daten für die Symbole erstellt und die Konsolenausgabe angegeben.
10	Es wird geprüft, ob der Befehl mit der Option --weather ausgeführt wurde.
11	Falls der Befehl Initialize mit der Option --weather ausgeführt wurde, wird programmatisch der Befehl aus dem weather Module ausgeführt.
12	Die Wetter-Anwendung wird konfiguriert und die benötigen Daten werden erstellt. Danach wird die Konsolenausgabe angegeben.
13	Falls keine Option ausgewählt wurde, wird in einer Konsolenausgabe angegeben, welche Optionen zur Verfügung stehen.

Tabelle 59 Realisierung: Initialize Skript Sequenzdiagramm

13.4. MVC (Django MTV) Komponente

Entsprechend dem ERD-Datenkonzept (siehe Abbildung [Konzept: Wetter ERD](#)) wurden folgende Modell-Klassen erstellt.

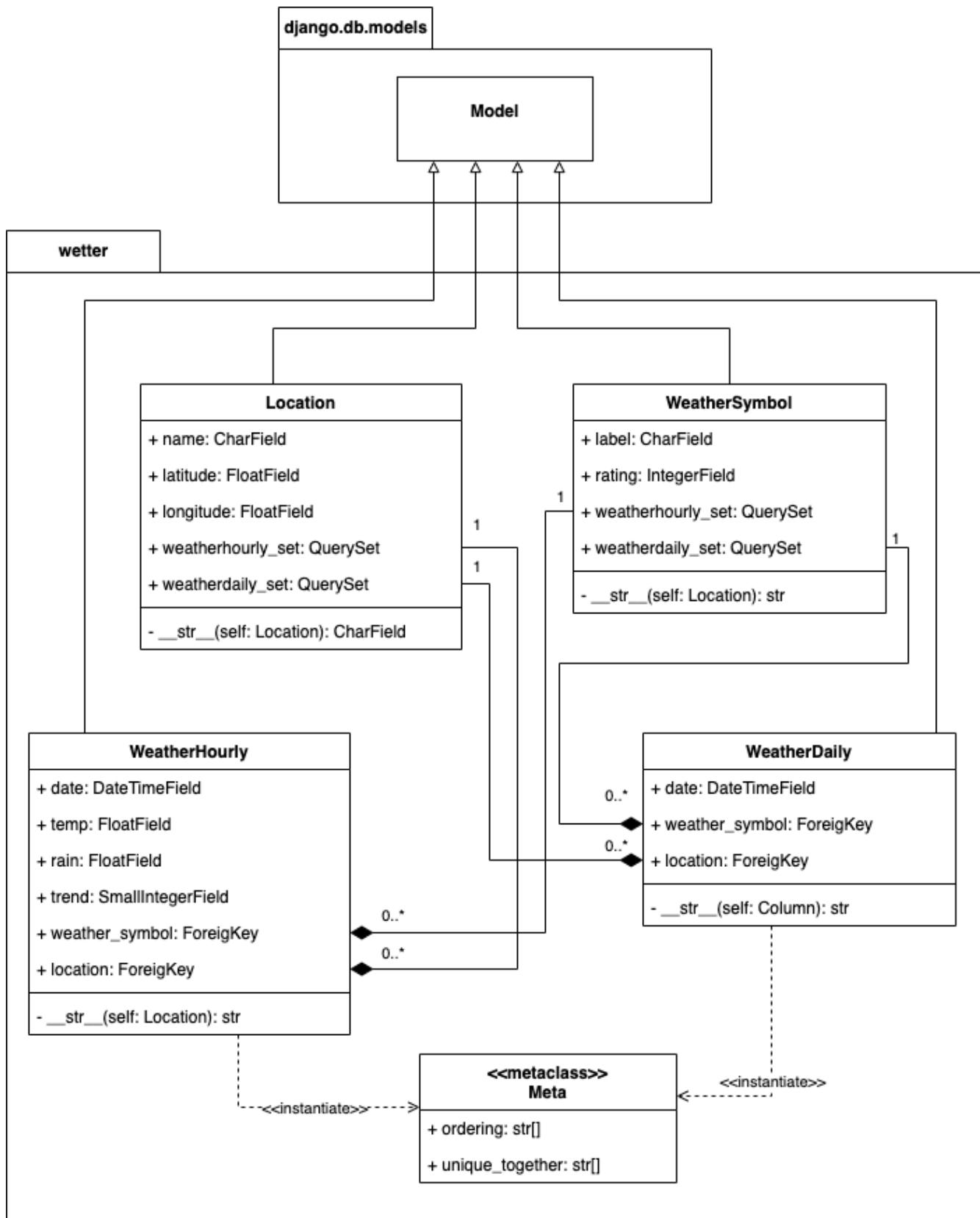


Abbildung 37 Realisierung: Klassendiagramm

Alle vier Klassen (Location, WeatherSymbol, WeatherHourly und WeatherDaily) sind von der Django Modell-Klasse abgeleitet. Wie in der [Initialisierung](#) bereits beschrieben, verfügt die Django Modell-Klasse über die ORM-Funktionalität. Darüber hinaus verwenden WeatherHourly und WeatherDaily die Meta-Klasse für die Konfiguration.

Die Beziehungen und die Multiplizität sind wie folgt:

Location und WeatherSymbol haben Beziehungen zu WeatherHourly und WeatherDaily in Form einer Komposition.

Die Multiplizität zu beiden ist gleich aufgebaut und lautet wie folgt:

Location-Objekte und WeatherSymbol-Objekte können keine oder mehrere WeatherHourly- und WeatherDaily-Objekte enthalten.

WeatherHourly-Objekte und WeatherDaily-Objekte sind abhängig von Location-Objekten und WeatherSymbol-Objekten und können nicht ohne Location-Objekte und WeatherSymbol-Objekte instanziert werden.

13.5. Routing

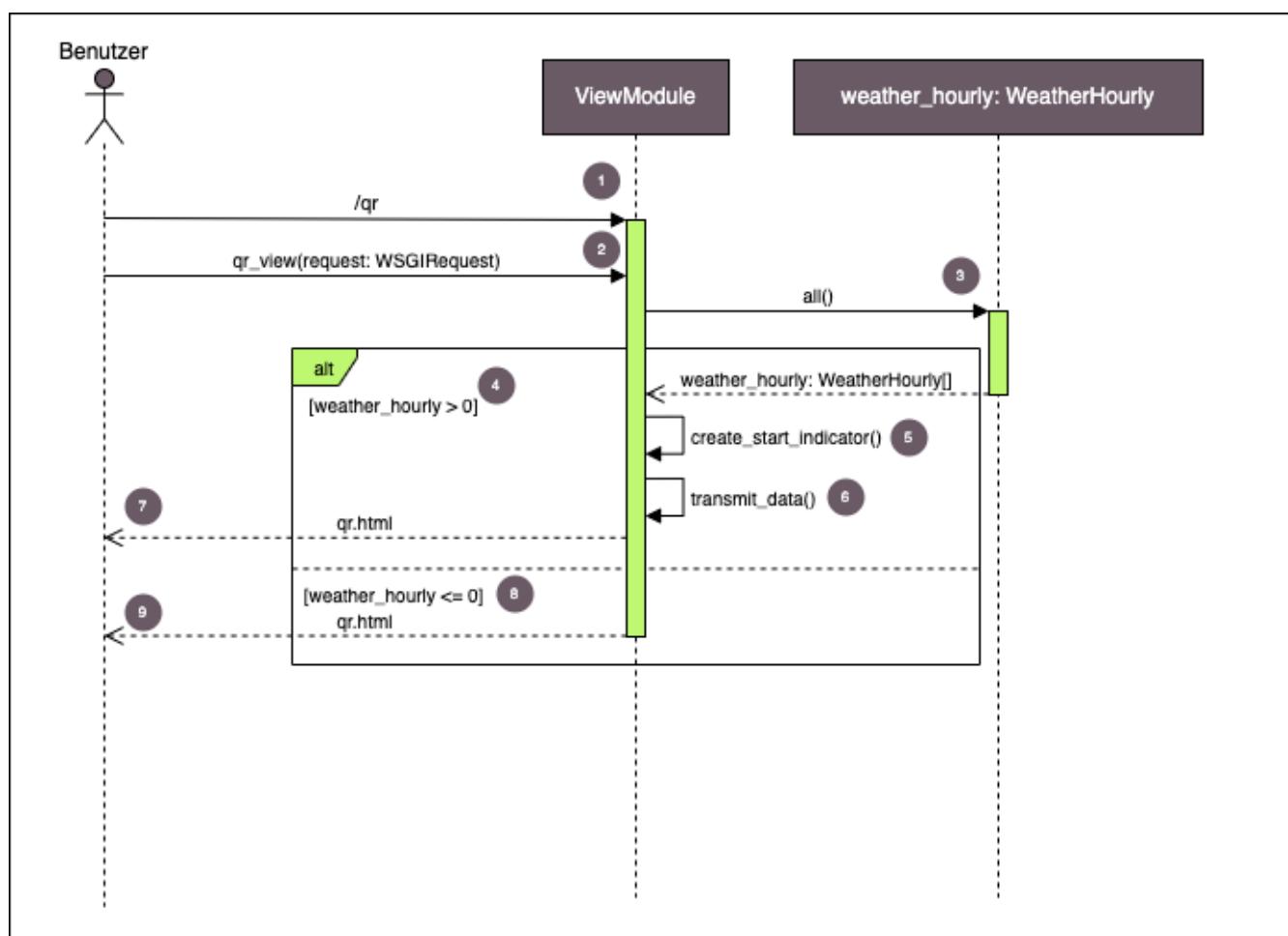


Abbildung 38 Realisierung: Sequenzdiagramm - Routing

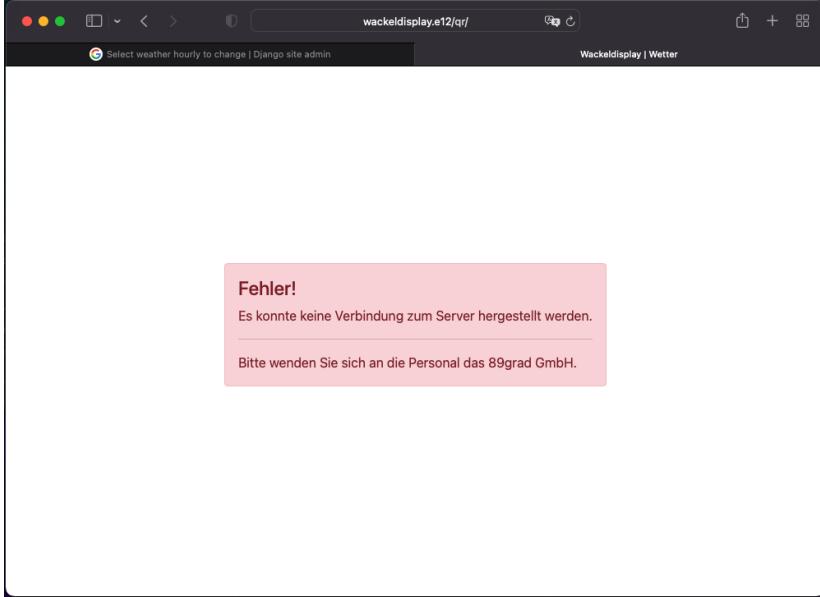
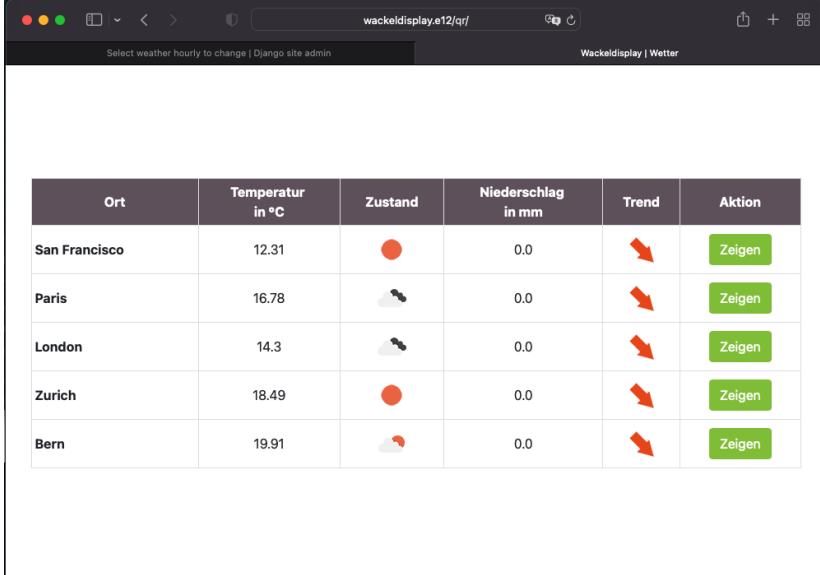
Nr.	Beschreibung
1, 2	Die URL-Auflösung wurde bereits im IST-Zustand beschrieben, daher werden diese Ereignisse hier lediglich abgekürzt wiedergegeben:

	Der Benutzer ruft /qr url auf.
3	Das ViewModule holt alle WeatherHourly-Einträge aus der Datenbank.
4	Kommt zur Anwendung, wenn Einträge in der Datenbank vorhanden sind.
5	Erstelle «Wackeldisplay ist gestartet Indikator».
6	Daten in eine Datei übertragen
7	Die HTML-Vorlage wird für die Anzeige der Daten in eine Tabelle geladen.
8	Andernfalls lade die Vorlage mit einer Meldung, dass keine Daten vorhanden sind.

Tabelle 60 Realisierung: Sequenzdiagramm – Routing

13.5.1. Templates

Für die Entwicklung der Vorlagen wurde das CSS-Framework Bootstrap Version 5 verwendet.

Bild	Beschreibung
	Wenn keine Daten angezeigt werden können oder ein Fehler auf dem Server vorliegt, wird die folgende Fehlermeldung angezeigt.
	Entsprechend, wie im Konzept, ist hier das Ergebnis des "Home Page" Seites. Der Code der Vorlage ist beigelegt. Siehe qr.html.

The screenshot shows a Django admin interface for a 'Wackeldisplay' application. The main page displays a table of weather data for five cities: San Francisco, Paris, London, Zurich, and Bern. Each row includes a temperature value, a weather icon, a trend indicator (a red downward arrow), and a 'Zeigen' button. A modal dialog titled 'Bestätigen' (Confirm) is overlaid on the page, asking 'Wollen Sie wirklich die Daten auf dem Wackeldisplay anzeigen lassen?' (Do you really want to display the data on the wackeldisplay?). It contains two buttons: 'Schliessen' (Close) and 'Bestätigen' (Confirm). The 'Bestätigen' button is highlighted with a green background.

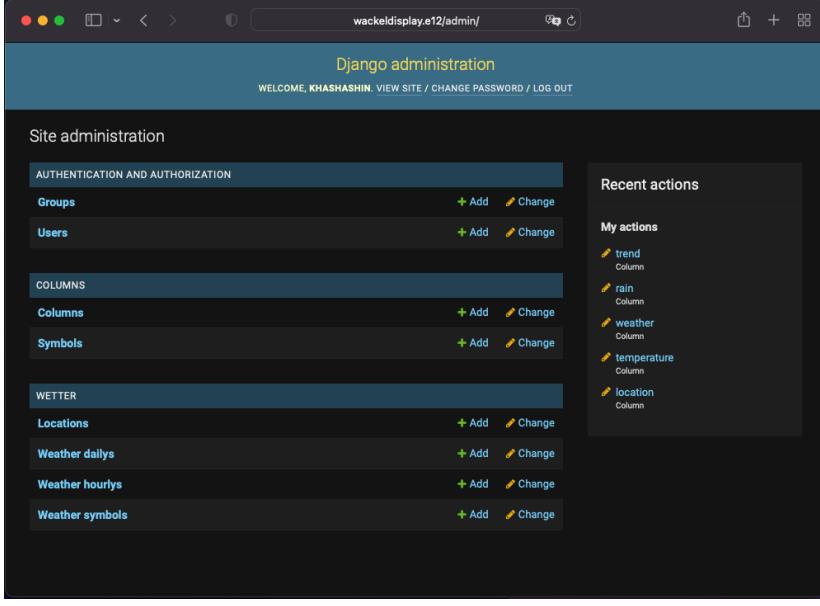
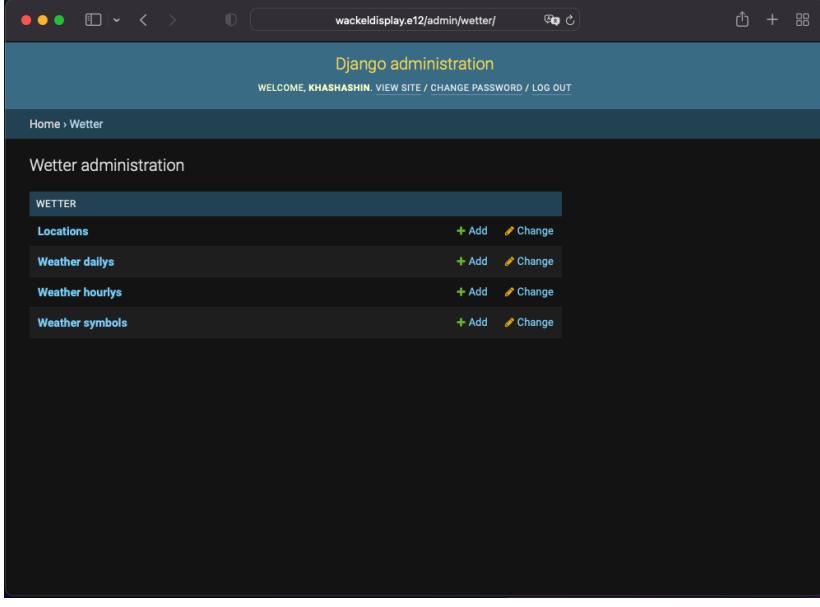
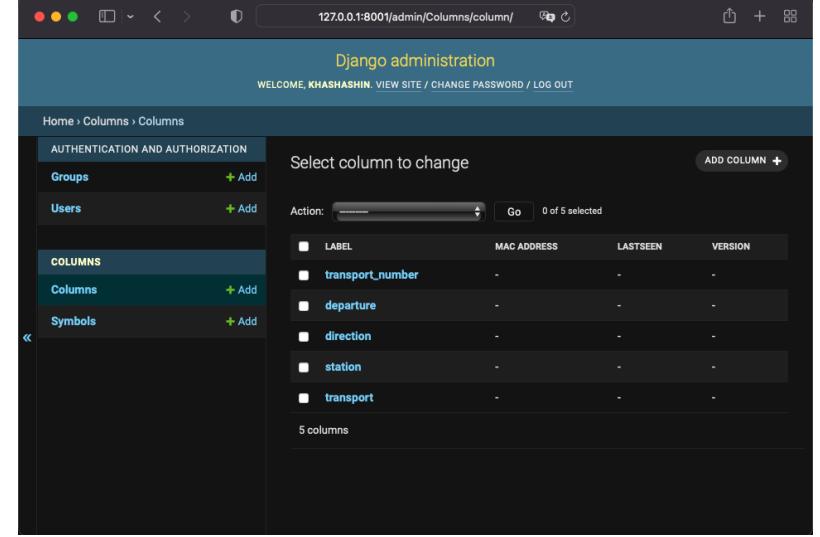
Wenn die Wackelanzeige **nicht** in Betrieb ist, wird die folgende «Alert» angezeigt. Das wird mittels JavaScript gehandelt. Siehe qr.html im Anhang.

The screenshot shows the same Django admin interface as the first one, but the modal dialog has changed. It now displays the message 'Wackeldisplay ist gerade in Betrieb, versuchen Sie es später noch einmal.' (The wackeldisplay is currently in operation, try again later). The 'Schliessen' button is visible at the bottom left of the dialog.

Wenn die Wackelanzeige in Betrieb ist, wird die folgende «Alert» angezeigt. Das wird mittels JavaScript gehandelt. Siehe qr.html im Anhang.

Tabelle 61 Realisierung: Templates Beschreibung

13.6. Datenverwaltung (Django-Admin)

Bild	Beschreibung
	Das ist die Übersicht aller Modelle, die man Konfiguriert hat in Django Admin anzuzeigen. Jedoch «Authentication and Autorisation» Modelle werden immer Standardmäßig vom Django selber erstellt. Neu wird hier Wetter Modelle angezeigt.
	Die Übersicht der Wetter Modelle.
	Die Übersicht der Locations Daten. Dieser Screenshot wurde nach der Ausführung der Initialize Kommando (Siehe Anhang initialize.py) gemacht und die Daten wurde automatisch erstellt.

Django administration

Add weather hourly

Date: Date: Today |

Time: Time: Now |

Temperature: Temperature in °C

Rain: Rain in mm

Trend: Trend of the Weather

Symbol:

Location:

Erstellung eines neuen Standortes.

Select weather symbol to change

SYMBOL	SYMBOL RATING
<input type="checkbox"/> snow	1
<input type="checkbox"/> thunderstorm	2
<input type="checkbox"/> shower rain	3
<input type="checkbox"/> rain	4
<input type="checkbox"/> broken clouds	5
<input type="checkbox"/> scattered clouds	6
<input type="checkbox"/> mist	7
<input type="checkbox"/> few clouds	8
<input type="checkbox"/> clear sky	9

FILTER

By Symbol rating

All
1
2
3
4
5
6
7
8
9

9 weather symbols

Die Übersicht der Symbol Daten.

Dieser Screenshot wurde nach der Ausführung der Initialize Kommando (Siehe Anhang initialize.py) gemacht und die Daten wurde automatisch erstellt.

Django administration

Add weather symbol

Symbol:

Symbol rating: 0 Symbol rating

Save and add another Save and continue editing **SAVE**

Erstellung eines neuen Symbols.

Select weather daily to change

DATE	SYMBOL	LOCATION
May 26, 2022, 1 p.m.	clear sky (9)	Paris
May 26, 2022, 1 p.m.	broken clouds (5)	London
May 26, 2022, 1 p.m.	clear sky (9)	Zurich
May 26, 2022, 1 p.m.	clear sky (9)	Bern
May 25, 2022, 10 p.m.	broken clouds (5)	San Francisco
May 25, 2022, 1 p.m.	broken clouds (5)	Paris
May 25, 2022, 1 p.m.	rain (4)	London
May 25, 2022, 1 p.m.	rain (4)	Zurich
May 25, 2022, 1 p.m.	broken clouds (5)	Bern
May 24, 2022, 10 p.m.	few clouds (8)	San Francisco
May 24, 2022, 1 p.m.	rain (4)	Paris
May 24, 2022, 1 p.m.	rain (4)	London
May 24, 2022, 1 p.m.	rain (4)	Zurich
May 24, 2022, 1 p.m.	rain (4)	Bern
May 23, 2022, 10 p.m.	broken clouds (5)	San Francisco

15 weather dailys

Die Übersicht der WeatherDaily Daten.

Dieser Screenshot wurde nach der Ausführung der Initialize Kommando (Siehe Anhang initialize.py) gemacht und die Daten wurde automatisch erstellt.

Add weather daily

Date: Today!

Time: Now

Symbol:

Location:

Save and add another Save and continue editing **SAVE**

Erstellung eines neuen Weather Daily Eintrags.

Select weather hourly to change Django site admin																																									
WELCOME KHASHASHIN VIEW SITE / CHANGE PASSWORD / LOG OUT																																									
Django administration																																									
Home > Wetter > Weather hourlys																																									
AUTENTICATION AND AUTHORIZATION																																									
Groups + Add																																									
Users + Add																																									
COLUMNS																																									
Columns + Add																																									
Symbols + Add																																									
WETTER																																									
Locations + Add																																									
Weather dailys + Add																																									
Weather hourlys + Add																																									
Weather symbols + Add																																									
Select weather hourly to change																																									
Action: <input type="button" value="Go"/> Go 0 of 5 selected																																									
<table border="1"> <thead> <tr> <th>LOCATION</th> <th>TEMPERATURE</th> <th>SYMBOL</th> <th>RAIN</th> <th>TREND</th> <th>DATE</th> </tr> </thead> <tbody> <tr> <td>San Francisco</td> <td>12.31</td> <td>clear sky (0)</td> <td>0.0</td> <td>2</td> <td>May 23, 2022, 8:46 a.m.</td> </tr> <tr> <td>Paris</td> <td>16.78</td> <td>broken clouds (5)</td> <td>0.0</td> <td>2</td> <td>May 23, 2022, 8:46 a.m.</td> </tr> <tr> <td>London</td> <td>14.3</td> <td>broken clouds (5)</td> <td>0.0</td> <td>2</td> <td>May 23, 2022, 8:46 a.m.</td> </tr> <tr> <td>Zurich</td> <td>18.49</td> <td>clear sky (0)</td> <td>0.0</td> <td>2</td> <td>May 23, 2022, 8:46 a.m.</td> </tr> <tr> <td>Bern</td> <td>19.91</td> <td>few clouds (6)</td> <td>0.0</td> <td>2</td> <td>May 23, 2022, 8:46 a.m.</td> </tr> </tbody> </table>						LOCATION	TEMPERATURE	SYMBOL	RAIN	TREND	DATE	San Francisco	12.31	clear sky (0)	0.0	2	May 23, 2022, 8:46 a.m.	Paris	16.78	broken clouds (5)	0.0	2	May 23, 2022, 8:46 a.m.	London	14.3	broken clouds (5)	0.0	2	May 23, 2022, 8:46 a.m.	Zurich	18.49	clear sky (0)	0.0	2	May 23, 2022, 8:46 a.m.	Bern	19.91	few clouds (6)	0.0	2	May 23, 2022, 8:46 a.m.
LOCATION	TEMPERATURE	SYMBOL	RAIN	TREND	DATE																																				
San Francisco	12.31	clear sky (0)	0.0	2	May 23, 2022, 8:46 a.m.																																				
Paris	16.78	broken clouds (5)	0.0	2	May 23, 2022, 8:46 a.m.																																				
London	14.3	broken clouds (5)	0.0	2	May 23, 2022, 8:46 a.m.																																				
Zurich	18.49	clear sky (0)	0.0	2	May 23, 2022, 8:46 a.m.																																				
Bern	19.91	few clouds (6)	0.0	2	May 23, 2022, 8:46 a.m.																																				
5 weather hourlys																																									
FILTER																																									
By Trend																																									
All																																									
2																																									
By Date																																									
Any date																																									
Today																																									
Past 7 days																																									
This month																																									
This year																																									

Die Übersicht der WeatherHourly Daten.

Dieser Screenshot wurde nach der Ausführung der Initialize Kommando (Siehe Anhang initialize.py) gemacht und die Daten wurde automatisch erstellt.

Add weather hourly Django site admin					
WELCOME KHASHASHIN VIEW SITE / CHANGE PASSWORD / LOG OUT					
Django administration					
Home > Wetter > Weather hourlys > Add weather hourly					
AUTENTICATION AND AUTHORIZATION					
Groups + Add					
Users + Add					
COLUMNS					
Columns + Add					
Symbols + Add					
WETTER					
Locations + Add					
Weather dailys + Add					
Weather hourlys + Add					
Weather symbols + Add					
Add weather hourly					
Date: <input calendar="" icon"="" type="text" value="Date: Today "/>					
Time: <input clock="" icon"="" type="text" value="Now "/>					
Date of the forecast					
Temperature: <input type="text" value="0"/> Temperature in °C					
Rain: <input type="text" value="0"/> Rain in mm					
Trend: <input type="text" value="0"/> Trend of the Weather					
Symbol: <input type="text" value="—"/> Weather symbol					
Location: <input type="text" value="—"/> Location of the forecast					
<input type="button" value="Save and add another"/> <input type="button" value="Save and continue editing"/> <input type="button" value="SAVE"/>					

Erstellung eines neuen Weather Hourly Eintrags.

Tabelle 62 Realisierung: Datenverwaltung - Django Admin

13.7. Wackeldisplay Worker (Python Skript)

Nachfolgend ist das Sequenzdiagramm des Wetter-Workers dargestellt. Um Platz zu sparen, ist es stark vereinfacht dargestellt.

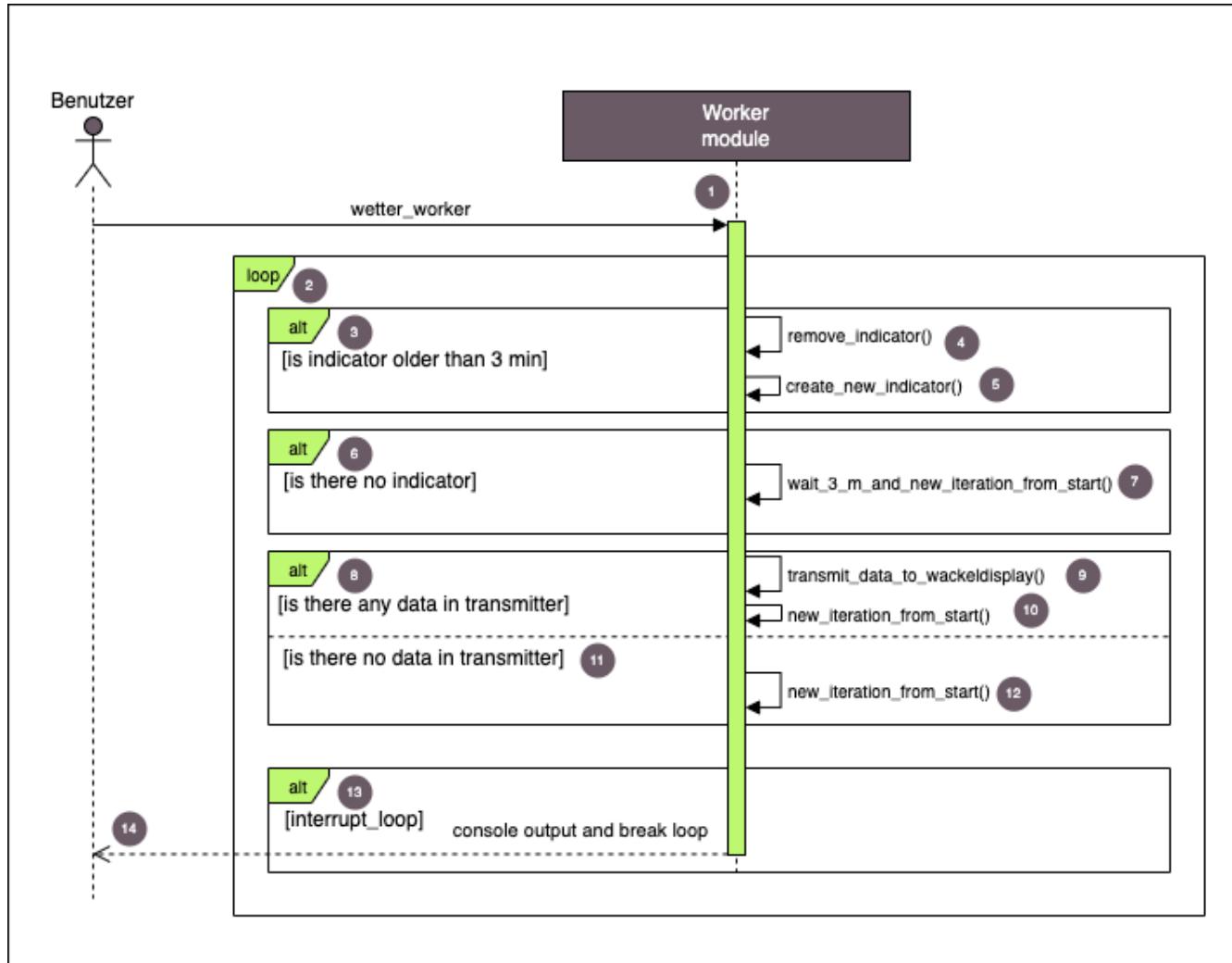


Abbildung 39 Realisierung: Sequenzdiagramm Wetter Worker - `wetter_worker.py`

Nr.	Beschreibung
1	Admin führt den Befehl <code>python manage.py wetter_worker</code> aus.
2	Es beginnt eine Endlosschleife.
3	Es wird geprüft, ob die Indikatordatei existiert und ob die Indikatordatei vor 3 Minuten erstellt wurde.
4	Wenn die Indikatordatei existiert und älter als 3 Minuten ist, wird die Indikatordatei gelöscht.
5	Danach wird die Indikator-Datei neu erstellt.
6	Wenn es keine Indikatordatei gibt oder die Indikatordatei nicht früher als 3 Minuten erstellt wurde.
7	Warte 3 Sekunden und starte die Schleife erneut.

8	Es wird geprüft, ob die Transmitterdatei existiert und ob die Transmitterdatei Daten enthält.
9	Wenn die Daten verfügbar sind, werden sie an Wackeldisplay gesendet und angezeigt.
10	Danach wird die Schleife neu gestartet.
11	Wenn die Transmitterdatei nicht vorhanden ist oder keine Daten darin enthalten sind.
12	Die Schleife wird neu gestartet.
13	Wenn der Administrator die Schleife mit einer Tastenkombination unterbricht.
14	Eine Konsolenausgabe wird ausgegeben und der Worker wird gestoppt.

Tabelle 63 Realisierung: Worker - *wetter_worker.py*

14. Einführung

Einführung ist sehr ähnlich zu dem, was im IST-Zustand beschrieben wurde. Die Software wird auf demselben Server installiert. Es wird unter dem Heimatverzeichnis des Fahrplanbenutzers in einem wd-weather Verzeichnis (entsprechend dem Namen des Repositorys) abgelegt.

14.1. Installation (Deployment)

Teilweise wurde dieser Prozess auch im IST-Zustand beschrieben. Darüber hinaus wurde auch die Datei README.md aktualisiert (siehe README.md im Anhang).

Nach der Einführung in die Hilfe folgen die Skripte zur automatischen Datenerstellung (siehe [Automatisation und Migration Skripte](#)). Nachdem die Daten angelegt sind, müssen zwei Cronjobs erstellt werden, die die Dateien pull_daily.py und pull_hourly.py steuern (dieser Schritt ist nur einmal notwendig).

14.2. Benutzeranleitung

Es gibt zwei Arten von Anleitungen, eine für die Entwickler (siehe README.md) und eine für die Benutzer des Wackeldisplays (siehe im Benutzerhandbuch⁸ im Anhang).

⁸ Das Kapitel Benutzerhandbuch wurde extra ohne Nummerierung erstellt, um das Ausdrucken zu erleichtern. Es ist für die Anwender des Wackeldisplays vorgesehen. Zudem wurde die Seitennummer in der Fusszeile ausgeblendet.

15. Abbildungsverzeichnis

Abbildung 1 Foto von Pedro Mirano auf Unsplash -----	0
Abbildung 2 89grad GmbH auf der Karte -----	7
Abbildung 3 89grad GmbH DNS -----	11
Abbildung 4 Arbeitsplatz von vorne -----	13
Abbildung 5 Arbeitsplatz von hinten -----	14
Abbildung 6 Laptop Konfiguration -----	15
Abbildung 7 Dateistruktur -----	16
Abbildung 8 Tägliche Backups Ordner Struktur -----	16
Abbildung 9 Übersicht der GitLab IPA-Dokumentation -----	17
Abbildung 10 GitLab IPA-Dokumentation Tags -----	17
Abbildung 11 OwnCloud Daten Version-----	18
Abbildung 12 Git CLI Wiederherstellung nach einem Tag -----	19
Abbildung 13 Gesamtübersicht des Projektplans-----	20
Abbildung 14 HERMES-gibb Projektvorgehensmodell-----	21
Abbildung 15 Projekt Organigramm-----	24
Abbildung 16 IST-Zustand: Systemarchitektur -----	49
Abbildung 17 IST-Zustand: Django Projekt Architektur -----	49
Abbildung 18 IST-Zustand: Wackeldisplay Kommunikation mit Django Server -----	51
Abbildung 19 IST-Zustand: SQLite Datenbank Datei -----	51
Abbildung 20 IST-Zustand: Wackeldisplay ERD-----	51
Abbildung 21 IST-Zustand: Physischer Datenbankzustand -----	52
Abbildung 22 IST-Zustand: Klassendiagramm-----	54
Abbildung 23 IST-Zustand: Fahrplaner Home -----	55
Abbildung 24 IST-Zustand: Django Server Anwendungsfälle -----	56
Abbildung 25 IST-Zustand: Sequenzdiagramm /fahrplaner Seite-----	60
Abbildung 26 Niederschlag pro Tag geteilt durch Stunde der letzten 12 Monate-----	78
Abbildung 27 Niederschlag: Historische Daten in 10 Jahren -----	79
Abbildung 28 Niederschlagsänderung in Letzte 12 Monaten-----	79
Abbildung 29 Konzept: Anwendungsfälle-----	81
Abbildung 30 Konzept: Django ERD-----	83
Abbildung 31 Konzept: Wetter ERD -----	84
Abbildung 32 Das "Home Page" Designkonzept-----	85
Abbildung 33 Konzept: Wireframe Daten zeigen-----	86
Abbildung 34 Konzept: Wireframe Daten zeigen. Bestätigen -----	86
Abbildung 35 Konzept: Django Projekt Architektur vor und nach -----	88
Abbildung 36 Realisierung: Initialize Skripte Sequenzdiagramm -----	94
Abbildung 38 Realisierung: Klassendiagramm -----	96
Abbildung 39 Realisierung: Sequenzdiagramm - Routing -----	97
Abbildung 40 Realisierung: Sequenzdiagramm Wetter Worker - wetter_worker.py-----	104
Abbildung 41 Design Guidelines: Farbpalette -----	131
Abbildung 42 Anhang: 89grad GmbH Dokumentenvorlage -----	135
Abbildung 43 Anhang: Dokumentenablage und Ordnerstruktur -----	140

16. Tabellenverzeichnis

Tabelle 1 Projektangaben gemäss Pkorg -----	1
Tabelle 2 Projektstandards -----	10
Tabelle 3 Projekt Software Liste-----	13
Tabelle 4 git-CLI Log Formatierung-----	18
Tabelle 5 HERMES-gibb Szenario -----	21
Tabelle 6 HERMES-gibb Phasen -----	22
Tabelle 7 HERMES-gibb Module -----	23
Tabelle 8 Projekt Meilensteine -----	23
Tabelle 9 Projektrollen -----	25
Tabelle 10 Arbeitsjournal Tag 1 -----	29
Tabelle 11 Arbeitsjournal Tag 2 -----	31
Tabelle 12 Arbeitsjournal Tag 3 -----	32
Tabelle 13 Arbeitsjournal Tag 4 -----	34
Tabelle 14 Arbeitsjournal Tag 5 -----	36
Tabelle 15 Arbeitsjournal Tag 6 -----	38
Tabelle 16 Arbeitsjournal Tag 7 -----	40
Tabelle 17 Arbeitsjournal Tag 8 -----	41
Tabelle 18 Arbeitsjournal Tag 9 -----	43
Tabelle 19 Arbeitsjournal Tag 10-----	44
Tabelle 20 IST-Zustand: Django Projekt Beschreibung -----	50
Tabelle 21 IST-Zustand: Wackeldisplay Beschreibung -----	50
Tabelle 22 IST-Zustand: Datenbank Tabellen Beschreibung-----	52
Tabelle 23 IST-Zustand: MVC und MTV Vergleich-----	53
Tabelle 24 IST-Zustand: Fahrplaner Screenshots-----	56
Tabelle 25 IST-Zustand Anwendungsfälle-----	59
Tabelle 26 IST-Zustand: /fahrplaner Seite Sequenzdiagramm Beschreibung-----	61
Tabelle 27 IST-Zustand: Vorhandene Routing -----	61
Tabelle 28 IST-Zustand: Django Admin Übersicht-----	63
Tabelle 29 IST-Zustand: Projektstruktur -----	64
Tabelle 30 IST-Zustand: Projekt Versionierung-----	66
Tabelle 31 Erste Einführungsschritte -----	68
Tabelle 32 Softwareupdate Schritte -----	68
Tabelle 33 IST-Zustand: Schwächen -----	69
Tabelle 34 Projektziele -----	70
Tabelle 35 Funktionale Anforderungen-----	71
Tabelle 36 Nicht Funktionale Anforderungen -----	71
Tabelle 37 Projekt Risikoanalyse -----	72
Tabelle 38 Legende: Schadensausmass -----	72
Tabelle 39 Legende: Eintrittswahrscheinlichkeit-----	72
Tabelle 40 Risikomatrix -----	72
Tabelle 41 Risikomatrix Beschriftungen-----	72
Tabelle 42 Risikograph -----	73
Tabelle 43 Variantenvergleich API-Quellen-----	75

Tabelle 44 Bewertungsschema-----	75
Tabelle 45 Konzept: Wackeldisplay Symbole -----	77
Tabelle 46 Konzept: Grenzwerte - Beschreibung der Wetter Symbole-----	80
Tabelle 47 Konzept: Formel für die Trendberechnung -----	81
Tabelle 48 Konzept: Wackeldisplay Anwendungsfall -----	82
Tabelle 49 Konzept: Django Datenbank Beschreibung-----	83
Tabelle 50 Konzept: Wetter ERD Beschreibung-----	85
Tabelle 51 Konzept: Symbolen Beschreibung-----	87
Tabelle 52 Testfälle: FT 1 QR-Code wird als einen Link erkannt -----	89
Tabelle 53 Testfälle: FT 2 Wackeldisplay startet -----	90
Tabelle 54 Testfälle: FT 3 Daten werden auf der Webseite angezeigt -----	90
Tabelle 55 Testfälle: FT 4 Bestimmte Daten werden auf der Wackeldisplay angezeigt -----	90
Tabelle 56 Testfälle: FT 5 Bestimmte Daten werden auf der Wackeldisplay nicht angezeigt -----	91
Tabelle 57 Realisierung: Wackeldisplay-----	92
Tabelle 58 Realisierung: Projekt Struktur-----	94
Tabelle 59 Realisierung: Initialize Skript Sequenzdiagramm -----	95
Tabelle 60 Realisierung: Sequenzdiagramm – Routing-----	98
Tabelle 61 Realisierung: Templates Beschreibung-----	99
Tabelle 62 Realisierung: Datenverwaltung - Django Admin-----	103
Tabelle 63 Realisierung: Worker - wetter_worker.py-----	105
Tabelle 64 Projektinitialisierungsauftrag-----	119
Tabelle 65 Projektfreigabe-----	119
Tabelle 66 Phasenfreigabe (Realisierung)-----	120
Tabelle 67 Phasenfreigabe (Einführung)-----	120
Tabelle 68 Sitzungsprotokoll 1-----	121
Tabelle 69 Sitzungsprotokoll 2-----	123
Tabelle 70 Sitzungsprotokoll 3-----	124
Tabelle 71 Sitzungsprotokoll 4-----	125
Tabelle 72 Sitzungsprotokoll 5-----	125
Tabelle 73 Sitzungsprotokoll 6-----	126
Tabelle 74 Sitzungsprotokoll 7-----	126
Tabelle 75 Sitzungsprotokoll 8-----	127
Tabelle 76 Sitzungsprotokoll 9-----	128
Tabelle 77 Sitzungsprotokoll 10-----	128
Tabelle 78 Sitzungsprotokoll 11-----	129
Tabelle 79 Design Guidelines: Farbpalette -----	131
Tabelle 80 Design Guidelines: Farbtöne-----	132
Tabelle 81 Testprotokoll: T-01 -----	132
Tabelle 82 Testprotokoll: T-02 -----	133
Tabelle 83 Testprotokoll: T03 -----	133
Tabelle 84 Testprotokoll: T-04 -----	134
Tabelle 85 Testprotokoll: T-05 -----	134
Tabelle 86 Anhang: Software-Versionsverwaltung-----	139
Tabelle 87 Dateien, die zu ignorieren sind -----	141
Tabelle 88 Anhang: Readme - README.md -----	142

Tabelle 89 Anhang: Skripte - pull_hourly.py-----	144
Tabelle 90 Anhang: Skripte - pull_daily.py -----	145
Tabelle 91 Anhang: Skripte - wetter_worker.py-----	149
Tabelle 92 Anhang: Skripte - Initialize.py-----	153
Tabelle 93 Anhang: Skripte - initiate_locations.py-----	154
Tabelle 94 Anhang: Skripte - initiate_wstates.py-----	155
Tabelle 95 Anhang: Skripte - 1_get_headers.py-----	156
Tabelle 96 Anhang: Skripte - 2_get_daily_precipitation_data.py-----	157
Tabelle 97 Anhang: Skripte - 3_group_data_by_precipitation.py-----	158
Tabelle 98 Anhang: Skripte - 4_count_similarities.py-----	158
Tabelle 99 Anhang: Skripte - constants.py-----	159
Tabelle 100 Anhang: Routing - core/urls.py-----	159
Tabelle 101 Anhang: Routing - wetter/urls.py-----	159
Tabelle 102 Anhang: Django Admin - admin.py-----	161
Tabelle 103 Anhang: Models - models.py-----	163
Tabelle 104 Anhang: Templates - base.html-----	164
Tabelle 105 Anhang: Templates - wetter/qr.html-----	167
Tabelle 106 Anhang: Views - views.py-----	168
Tabelle 107 Anhang: Hilfeskripte - constants.py-----	169
Tabelle 108 Anhang: Hilfeskripte - ow_api_service.py-----	169
Tabelle 109 Anhang: Hilfeskripte - manage.py-----	170
Tabelle 110 Anhang: Django Settings - settings-devel.py-----	174
Tabelle 111 Anhang: Tests - tests.py-----	179
Tabelle 112 Anhang: Testdata - test_data.json-----	186
Tabelle 113 Anhang: Testdata - corrupted_data.json-----	192

17. Literatur- und Quellenverzeichnis

Git Tower, "To checkout a specific commit, you can use the git checkout command and provide the revision hash as a parameter: git checkout 757c47d4," [Online]. Available: <https://www.git-tower.com/learn/git/faq/git-checkout-commits>.

HERMES, «Die Initialisierung schafft eine definierte Ausgangslage für das Projekt und stellt sicher...» [Online]. Available: <https://www.hermes.admin.ch/de/projektmanagement/verstehen/phasen-und-meilensteine/initialisierung.html>.

Stackoverflow, «Set the desired message off by setting the config value to false:,» [Online]. Available: <https://stackoverflow.com/questions/36794501/disable-warning-about-detached-head>.

K. Sudhakaran, «A tag can be created using the git tag command.,» [Online]. Available: <https://www.tutorialspoint.com/how-to-tag-a-commit-in-git>.

Stackoverflow, «How to split terminal into panes in macbook?,» [Online]. Available: <https://stackoverflow.com/questions/51110685/how-to-split-terminal-into-panes-in-macbook>.

OwnCloud, «Versions are displayed in the WebUI TA \l "UI: User Interface" \s "UI" \c 15 in the details view in the right sidebar,» [Online]. Available: https://doc.owncloud.com/server/next/admin_manual/configuration/files/file_versioning.html.

C. Gu, «I'd like you to see if the result goes better if you create a custom A3 paper size (297x420 mm) with no margins.,» [Online]. Available: <https://answers.microsoft.com/en-us/msoffice/forum/all/a3-printing-in-excel/ef3c13f3-dcd9-45cc-905c-ee22e28379d2>.

HERMES, «Module sind wiederverwendbare Bausteine zur Erstellung von Szenarien. Ein Szenario besteht aus mehreren Modulen.,» [Online]. Available: <https://www.hermes.admin.ch/de/projektmanagement/verstehen/module.html>.

A. Müller, «IPA-Dokumentenvorgabe_2022_3_1.pdf,» ICT-Berufsbildung, Bern, 2022.

89grad GmbH, «Was bietet 89grad GmbH,» [Online]. Available: <https://www.89grad.ch/softwareloesungen-kundenportale>.

SCHERLER Taining & Projektmanagement, «Grundverständnis zum Aufbau der Methode und den Elementen von HERMES 5.1,» [Online]. Available: <https://www.hermes-projektmanagement-training.ch/hilfsmittel>.

HERMES, «HERMES definiert ein Rollenmodell und beschreibt standardisierte Rollen, um ein einheitliches...» [Online]. Available: <https://www.hermes.admin.ch/de/projektmanagement/verstehen/rollen.html>.

CREATEY, «What are Network Diagrams? it is a visual representation of a cluster or a small structure of networking devices.,» [Online]. Available: <https://creately.com/blog/diagrams/network-diagram-guide-tutorial/>.

C. Hogan, «The Top 9 Weather APIs for 2022,» [Online]. Available: <https://www.tomorrow.io/blog/top-8-weather-apis-for-2022/>.

RapidApi, «Top 8 Best Free Weather APIs to Access Global Weather Data (Updated for 2022),» [Online]. Available: <https://web.archive.org/web/20220407011409/https://rapidapi.com/blog/access-global-weather-data-with-these-weather-apis/>.

Stackoverflow, «A metaclass is drawn using the class notation plus the <<metaclass>> stereotype.,» [Online]. Available: <https://stackoverflow.com/questions/1483273/how-to-draw-a-classs-metaclass-in-uml>.

Stackoverflow, «If you want to indicate that something is not for public use, but it should act like protected use _,» [Online]. Available: <https://stackoverflow.com/questions/1301346/what-is-the-meaning-of-single-and-double-underscore-before-an-object-name>.

Wikipedia, «Die Modifizierte Chen-Notation (Modified Chen Notation, MC-Notation) ist eine Erweiterung der Chen-Notation,» [Online]. Available: [https://de.wikipedia.org/wiki/Chen-Notation#Modifizierte_Chen-Notation_\(MC-Notation\)](https://de.wikipedia.org/wiki/Chen-Notation#Modifizierte_Chen-Notation_(MC-Notation)).

HERMES, «Jedes Phasenende ist durch einen Meilenstein bestimmt, der den Entscheid zum weiteren Vorgehen hervorhebt.,» [Online]. Available: <https://www.hermes.admin.ch/de/projektmanagement/verstehen/phasen-und-meilensteine.html>.

Wikipedia, «Der Begriff White-Box-Test (seltener auch Glass-Box-Test) bezeichnet eine Methode des Software XE "Software:Software ist eine Sammlung von Anweisungen, die einem Computer sagen, wie er funktionieren soll." -Tests, bei der die Tests mit Kenntnissen über die innere Funktionsweise des zu testenden Systems entwickelt werden.,» [Online]. Available: <https://de.wikipedia.org/wiki/White-Box-Test>.

W. McMullin, «So schreiben Sie Testfälle für Software XE "Software:Software ist eine Sammlung von Anweisungen, die einem Computer sagen, wie er funktionieren soll." : Beispiele und Lernprogramm,» [Online]. Available: <https://de.parasoft.com/blog/how-to-write-test-cases-for-software-examples-tutorial>.

OfficeBeginner, «Click on the Settings -> Print All Pages. A drop-down menu appears. At the bottom of the menu, simply uncheck the Print Markup option.,» [Online]. Available: <https://officebeginner.com/msword/how-to-print-a-word-document-without-comments/#:~:text=Click%20on%20the%20Settings%20%2D%3E%20Print,comments%20from%20the%20print%20view>.

Microsoft, «Print a document in Word for Mac,» [Online]. Available:
<https://support.microsoft.com/en-us/office/print-a-document-in-word-for-mac-2d92b498-01a2-4a88-b833-83027173ae9c>.

SRF, «Temperatur und Niederschlag,» [Online]. Available:
<https://www.srf.ch/meteo/wetter/Bern/46.9471,7.4441?geolocationNameId=b37f9339c3e89d6ef433634ef14933b6>.

IBM, «Monthly Weather,» [Online]. Available:
<https://weather.com/weather/monthly/l/d171b86d9013f975a1f32bd871beadeb741230ee1db7d4a182a9ea1a641d9e56>.

Bundesamt für Meteorologie und Klimatologie MeteoSchweiz, «Jahresverlauf Temperatur, Sonne, Niederschlag,» [Online]. Available:
<https://www.meteoschweiz.admin.ch/home/klima/klima-der-schweiz/jahresverlauf-temperatur-sonne-niederschlag.html>.

wetter.com, «Klimadaten Wetterstation Bern,» [Online]. Available:
<https://www.wetter.com/reise/klima/klimatabelle/schweiz-bern-CH0CH0324.html>.

Git, «There are several built-in formats, and you can define additional formats by setting a pretty.<name> config option,» [Online]. Available: <https://git-scm.com/docs/pretty-formats>.

Wikipedia, «Model–view–controller (MVC) is a software design pattern[1] commonly used for developing user interfaces that divide the related program logic into three interconnected elements.,» [Online]. Available:
<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>.

Stackoverflow, «Quick and easiest solution: kill -9 \$(lsof -ti:3000),» [Online]. Available:
<https://stackoverflow.com/questions/3855127/find-and-kill-process-locking-port-3000-on-mac>.

V. Gite, «Type any one of the following command to find os name and version in Linux: cat /etc/os-release,» [Online]. Available: <https://www.cyberciti.biz/faq/how-to-check-os-version-in-linux-command-line/>.

Stackoverflow, «Is there a naming convention for Django XE "Django:Ein freies und kostenloses, in Python geschriebenes Webanwendungs-Framework." apps? They must be valid package names. That rules out 2 ("import my-django-app" would be a syntax error),» [Online]. Available: <https://stackoverflow.com/questions/3098681/is-there-a-naming-convention-for-django-apps>.

U. Özyilmazel, «Django XE "Django:Ein freies und kostenloses, in Python geschriebenes Webanwendungs-Framework." Model Best Practices,» [Online]. Available:
<https://dev.to/vigo/django-model-best-practices-3e8e>.

Django XE "Django:Ein freies und kostenloses, in Python geschriebenes Webanwendungs-Framework." Project, «A convenience method for looking up an object with the given kwargs (may be empty if your model has defaults for all fields), creating one if necessary.,» [Online]. Available:
<https://docs.djangoproject.com/en/4.0/ref/models/querysets/#get-or-create>.

Stackoverflow, «don't use the python datetime functions when getting today/now. Use: from django.utils import timezone,» [Online]. Available:
<https://stackoverflow.com/questions/70185513/runtimewarning-datetimefield-model-date-received-a-naive-datetime-while-time-zo>.

Stackoverflow, «You can make use of either unittest.TestCase.tearDown or unittest.TestCase.tearDownClass,» [Online]. Available:
<https://stackoverflow.com/questions/64072833/does-django-testing-have-a-breakdown-similar-to-setup-that-is-ran-after-all>.

Stackoverflow, «What I always do is use pragma: no cover for this which you already have in your exclude_lines.,» [Online]. Available:
<https://stackoverflow.com/questions/27064807/python-coverage-py-exclude-lines>.

A. Johnson, «Getting a Django XE "Django:Ein freies und kostenloses, in Python geschriebenes Webanwendungs-Framework." Application to 100% Test Coverage,» [Online]. Available: <https://adamj.eu/tech/2019/04/30/getting-a-django-application-to-100-percent-coverage/>.

Stackoverflow, «Your problem is that you are calling the get method - which raises the exception - before it is passed to assertRaises.,» [Online]. Available:
<https://stackoverflow.com/questions/11109468/how-do-i-import-the-django-doesnotexist-exception>.

P. Doblhofer, «For semantic versioning, a version number consists mainly of 3 digits: MAJOR.MINOR.PATCH.,» [Online]. Available: <https://www.philipp-doblhofer.at/en/blog/automatic-changelog-and-versioning-with-git/>.

18. Abkürzungverzeichnis

AJAX: Asynchronous JavaScript and XML -----	50
API: Application Programming Interface -----	8, 9, 32, 40, 73, 75, 84, 88
BBT: Berufsbildung und Technologie -----	27
CI: continuous integration -----	38, 42
CLI: command-line interface -----	17, 18, 28, 50, 64, 65
CSV: Comma-Separated Values -----	75
E2E: End to end -----	9
ERD: Entity Relationship Diagram -----	52, 53, 82, 83, 84, 85, 96, 98, 113
gibb: Gewerblich-industrielle Berufsschule Bern-----	1, 9, 10, 20, 21, 22, 28, 29, 30, 50, 109
GmbH: Gesellschaft mit beschränkter Haftung--	1, 7, 9, 10, 11, 12, 14, 15, 17, 27, 30, 48, 49, 58, 59,
	60, 67, 77, 78, 81, 82, 86, 88, 89, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 121
GUI: graphical user interface-----	28
HASH: Hashfunktion -----	17
HERMES: Handbuch der Elektronischen Rechenzentren des Bundes, eine Methode zur Entwicklung von Systemen -----	1, 9, 10, 20, 21, 22, 24, 28, 29, 30, 50, 109
HTML: HyperText Markup Language -----	9, 42, 43, 48, 54, 72
IDE: integrated development environment -----	13, 39
IoT: Internet of Things -----	8, 9
IPA: Individuelle Praktische Arbeit----	2, 8, 9, 10, 11, 12, 13, 14, 16, 19, 22, 24, 27, 28, 30, 39, 48, 51,
	109, 110
ISDS: Informationssicherheit und Datenschutz-----	22, 32, 112
IT: Informationstechnologie -----	1, 19, 20, 21, 25, 48, 77
JS: JavaScript -----	9
JSON: JavaScript Object Notation -----	2, 70, 71, 72, 75
M2M: Many to many -----	83
MAC: Media-Access-Control-Address-----	2, 36, 53, 73
MQTT: Message Queuing Telemetry Transport-----	2, 51, 52
ORM: Object-relational mapping-----	54, 55
PO: Product Owner-----	8
POC: Proof of Concept -----	66
QR-Code: quick response code-----	44, 50, 56, 58, 59, 60, 61, 69, 81, 82, 89, 90, 91
SBB: Schweizerische Bundesbahnen-----	9
SSH: Secure Shell Protocol -----	11, 67
SSR: Server-side scripting -----	50
UI: User Interface -----	99
UML: Unified Modeling Language -----	32, 33
URL: Uniform Resource Locator -----	2, 56, 58, 59, 60, 61, 62, 82, 89, 90
USB: Universal Serial Bus-----	15, 17
WLAN: wireless local area network -----	51, 52, 77
XML: Extensible Markup Language -----	75
YAML: YAML Ain't Markup Language -----	75

19. Glossar

404 Fehlermeldung

Ein HTTP-Statuscode wird von einem Server auf jede HTTP-Anfrage als Antwort geliefert. 404 bedeutet "Die angeforderte Ressource wurde nicht gefunden.", 62

A3-Format

Die Norm basiert auf dem metrischen Messsystem, 28

Agiles Projektmanagement

Agiles Projektmanagement stellt die Rollen, Prozesse und Projektpläne des klassischen Vorgehens in Frage. Stattdessen wird der Schwerpunkt auf die intensive Einbeziehung der Beteiligten während des gesamten Projekts und die kontinuierliche Bereitstellung der Ergebnisse für sie gelegt., 48

Black-Box

Black-Box-Tests sind eine Methode des Softwaretests, bei der die Funktionalität einer Anwendung untersucht wird, ohne in ihre internen Strukturen oder Abläufe Einblick zu nehmen., 33, 43, 88

C++

Eine kompilierte, statisch typisierte Allzweck-Programmiersprache., 2, 48, 71, 89

Closed Source

Proprietäre Software, auch bekannt als Closed-Source-Software, ist Computersoftware, für die der Eigentümer oder eine andere Person Lizenzrechte behält, um die Software zu nutzen oder zu verändern., 48

Code Syntax

der Satz von Regeln, der die Kombinationen von Symbolen festlegt, die als korrekt strukturierte Aussagen oder Ausdrücke in dieser Sprache gelten., 34

Continuous Deployment

Der Prozess der automatischen Auslieferung der Software an die produktive Infrastruktur., 67

Continuous Integration

Der Prozess des kontinuierlichen Zusammenfügens von Komponenten zu einer Anwendung., 67, 89

Coverage

Ein Mass, das bei Softwaretests verwendet wird. Sie zeigt den Prozentsatz des Programmquellcodes an, der während des Tests ausgeführt wurde., 42, 43

Django

Ein freies und kostenloses, in Python geschriebenes Webanwendungs-Framework., 3, 5, 8, 9, 11, 22, 32, 35, 37, 38, 40, 42, 48, 50, 51, 52, 54, 55, 58, 59, 60, 61, 62, 63, 64, 65, 68, 75, 77, 81, 82, 83, 88, 92, 94, 96, 97, 98, 101, 102, 124

ESP8266

Der ESP8266 ist ein Mikrocontroller des chinesischen Herstellers Espressif Systems mit einer Wi-Fi-Schnittstelle., 73

Der ESP8266 ist ein Mikrocontroller des chinesischen Herstellers Espressif Systems mit einer Wi-Fi-Schnittstelle., 2, 11, 51, 52, 69

Der ESP8266 ist ein Mikrocontroller des chinesischen Herstellers Espressif Systems mit einer Wi-Fi-Schnittstelle., 77

Der ESP8266 ist ein Mikrocontroller des chinesischen Herstellers Espressif Systems mit einer Wi-Fi-Schnittstelle., 89

Exception

Ein Mechanismus von Programmiersprachen zur Beschreibung der Reaktion eines Programms auf Laufzeitfehler und andere mögliche Probleme (Ausnahmen), die auftreten können., 42

- Hosting
 - die Bereitstellung von Computerausrüstung und Software., 48
- JavaScript
 - eine Programmiersprache, wird am häufigsten in der Anwendungsentwicklung und in Browzern verwendet, um sie interaktiv und "lebendig" zu gestalten., 9, 48
- jinja-Template Engine
 - Jinja ist eine Web-Template-Engine für die Programmiersprache Python., 50
- Lean Design
 - Anwendung der Lean-Prinzipien im Kontext des Designs., 48
- Mockups
 - ein massstabsgerechtes Modell eines Entwurfs, das zu Lehrzwecken, zur Demonstration, zur Bewertung des Entwurfs und für andere Zwecke verwendet wird., 9, 33, 35
- Open Source
 - Offener Quellcode ist ein frei verfügbarer Programmcode, der verändert und weiterverbreitet werden kann., 10, 48
- Paket
 - Ein Paket in Python ist ein Verzeichnis, das andere Verzeichnisse und Module enthält, aber zusätzlich `__init__.py`. Pakete werden verwendet, um einen Namensraum zu bilden, in dem man mit Modulen arbeiten kann, die eine bestimmte Verschachtelungsebene (durch einen Punkt) aufweisen., 42, 50, 51, 64, 77, 88
- Pkorg
 - Als online-basiertes Informationssystem begleitet PkOrg den komplexen Ablauf des Qualifikationsverfahrens und unterstützt die Prüfungsorganisationen der beruflichen Grundbildung bei ihrer Arbeit., 1, 14, 30, 112, 113
- Port
 - Port ist ein Kommunikationsendpunkt. Auf der Softwareebene, innerhalb eines Betriebssystems, ist ein Port ein logisches Konstrukt, das einen bestimmten Prozess oder eine Art von Netzwerkdienst identifiziert., 38
- Proxy
 - Ein Proxy-Server ist eine Serveranwendung, die als Vermittler zwischen einem Client, der eine Ressource anfordert, und dem Server, der diese Ressource bereitstellt, fungiert., 40, 44
- Python
 - Eine universelle Hochsprachenprogrammierung mit dynamischer strikter Typisierung und automatischer Speicherverwaltung, die darauf ausgerichtet ist, die Produktivität der Entwickler zu erhöhen, die Lesbarkeit und Qualität des Codes zu verbessern und die Portabilität der in ihr geschriebenen Programme zu gewährleisten., 5, 8, 9, 13, 32, 48, 68, 75, 76, 94, 118, 124
- Release
 - Freigabe der endgültigen Version des Programms - ein einsatzbereites Produkt., 42
- Request
 - Die Nachricht, die von einem Client an einen Server gesendet wird, 39, 41
- Routing
 - Routing oder Router in der Webentwicklung ist ein Mechanismus, bei dem HTTP-Anfragen an den Code weitergeleitet werden, der sie bearbeitet., 37, 38, 62, 94, 124
- Schrittmotoren
 - Ein synchroner bürstenloser Elektromotor mit mehreren Wicklungen, 2
- Screen Session

Screen oder GNU Screen ist ein Terminal-Multiplexer. Man kann eine Screen-Session starten und dann eine beliebige Anzahl von Fenstern (virtuellen Terminals) innerhalb dieser Session öffnen., 68, 69

Skripte

eine Abfolge von Aktionen, die mit einer Skript-Programmiersprache (JavaScript, PHP, Perl, Python usw.) beschrieben werden, 9, 10, 40, 41

Software

Software ist eine Sammlung von Anweisungen, die einem Computer sagen, wie er funktionieren soll., 2, 3, 8, 11, 13, 48, 51, 67, 69, 71, 73, 77, 89, 97, 100

SQLite

ist eine in der Sprache C geschriebene Datenbank-Engine., 52, 83, 96

Terminal

Terminalemulator im Betriebssystem macOS von Apple, 28

UI

ist der Bereich, in dem Interaktionen zwischen Menschen und Maschinen stattfinden., 50

Unit Test

Unit Testing ist eine Softwaretestmethode, bei der einzelne Einheiten des Quellcodes auf ihre Gebrauchstauglichkeit getestet werden., 9

Use Case Diagramm

Ein Anwendungsfalldiagramm, auch Nutzungsfalldiagramm genannt, ist einer der Diagrammtypen der Unified Modeling Language (UML), 62

Virtuellen Umgebung

ein in sich geschlossener Verzeichnisbaum, der eine Python-Installation für eine bestimmte Python-Version sowie eine Reihe von zusätzlichen Paketen enthält., 68

Wackeldisplay

Ausrüstung entworfen von 89grad GmbH, 1, 2, 7, 8, 9, 11, 22, 35, 36, 39, 47, 50, 51, 52, 53, 55, 56, 57, 58, 59, 60, 62, 65, 69, 70, 71, 72, 77, 78, 81, 82, 83, 84, 85, 86, 88, 89, 90, 91, 92, 94, 110, 112, 113, 114, 115, 116, 117, 124

Web

Ein verteiltes System, das den Zugriff auf verknüpfte Dokumente ermöglicht, die sich auf verschiedenen, mit dem Internet verbundenen Computern befinden., 2, 8, 11, 48, 50, 54, 61, 62, 82

White-Box

Methode des Softwaretests, bei der nicht die Funktionalität, sondern die internen Strukturen oder Abläufe einer Anwendung getestet werden., 34, 88

Worker

Separater Kontext für Hintergrundaufgaben, 22, 37, 39, 40, 41, 59, 60, 67, 68, 69, 94, 124

20. Anhänge

20.1. Phasen Freigabe

Dieses Kapitel enthält die eingescannten Unterschriften des Auftraggebers und des Projektleiters.

20.1.1. Projektinitialisierungsauftrag

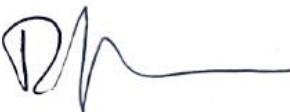
Bemerkungen	
Unterschrift Projektleiter	Unterschrift Auftraggeber
02.05.22 	

Tabelle 64 Projektinitialisierungsauftrag

20.1.2. Projektfreigabe

Bemerkungen	
Unterschrift Projektleiter	Unterschrift Auftraggeber
06.05.22 	

Tabelle 65 Projektfreigabe

20.1.3. Phasenfreigabe (Realisierung)

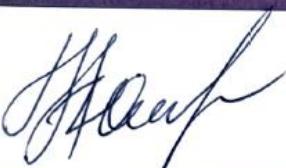
Bemerkungen	
Unterschrift Projektleiter	Unterschrift Auftraggeber
12.05.22 	

Tabelle 66 Phasenfreigabe (Realisierung)

20.1.4. Phasenfreigabe (Einführung)

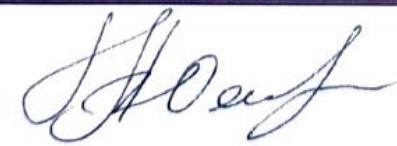
Bemerkungen	
Unterschrift Projektleiter	Unterschrift Auftraggeber
19.05.22 	

Tabelle 67 Phasenfreigabe (Einführung)

20.2. Sitzungsprotokolle

20.2.1. Sitzung 1 Datum: 02. Mai 2022

Thema	Daily Meeting	
Projekt	Wackeldisplay Wetter	
Anwesende	1. Yusup Khasbulatov 2. Ramun Hofmann	
Abwesende		
Sitzungsort	Eigerstrasse 12 89grad GmbH	
Protokollführer	Yusup Khasbulatov	
Traktanden	<ul style="list-style-type: none"> • Was wurde gemacht • Wo gibt es Probleme? • Was ist geplant 	
Ablauf		
<p>Wir haben meinen Zeitplan angeschaut und entschieden die Aufgaben zu gruppieren da es zu gross ist, um es auf einer A3-Seite auszudrucken. Ramun hat mir den Tipp gegeben, dass ich alle erreichten Kriterien-punkte mit einem Marker markieren sollte. Dies war ein sehr guter Tipp, da ich nun direkt sehe, was ich noch zu erledigen habe.</p>		
Beschlüsse und Aufgaben	Verantwortlich	
Alle nötigen Fragen für Expertenbesuch notieren.	Yusup Khasbulatov	

Tabelle 68 Sitzungsprotokoll 1

20.2.2. Sitzung 2 Datum: 05. Mai 2022

Thema	Expertenbesuch	
Projekt	Wackeldisplay Wetter	
Anwesende	1. Hans Engler 2. Ramun Hofmann 3. Florian Baumgartner 4. Yusup Khasbulatov	
Abwesende	1. Georg Achermann	
Sitzungsort	Eigerstrasse 12 89grad GmbH	
Protokollführer	Yusup Khasbulatov	
Traktanden	<ul style="list-style-type: none"> • Zeitplan Verbesserungsvorschläge <ul style="list-style-type: none"> ○ Wie wird berücksichtigt, dass die Aufgaben generalisiert aufgeschrieben werden (aus Platzgründen gruppiert)? ○ Wie sieht der bestehende Zeitplan aus, der am Montag erstellt wurde? ○ Entspricht er den geforderten Kriterien? ○ Wie lautet die Meinung von Herrn Engler zu den Meilensteinen? 	

- Zweiter Expertenbesuch verschieben
 - Der zweite Expertenbesuch ist zu nah. Daher wäre es gut ihn zu verschieben.
- Besprechung der folgenden Punkte betreffend die Organisation:
 - Dateiablage
 - Datei Wiederherstellung
 - Decken die bisherigen Vorbereitungen das Kriterium «Organisation des IPA Ergebnisse» ab?
 - Erwähnen, dass die Korrekturen durch Herr Guggisberg in einer Art «Review» gemacht werden und dass er als Qualitätsmanager in den Organigrammen eingetragen wird.
- Technische Risikoanalyse
 - Dieser Kapitel wird gemäss HERMES-gibb in der Phase Initialisierung geschrieben. Darf es daher aus der Dokumentation entfernt werden?

Beschlüsse

Präsentation:

- Präsentation muss 15-20 Minuten lang sein.

Fachgespräch:

- Erste Frage im Themenkomplex darf von 89grad GmbH gestellt werden.

Zeitplan:

- Die Aufgaben können gruppiert werden, um Platz zu sparen.
- Die Tätigkeiten im Arbeitsjournal sollen sich auf die gruppierte Aufgabe in dem Zeitplan beziehen.
- Der bestehende Zeitplan ist gut dargestellt und muss nicht angepasst werden.
- Meilensteine können angepasst werden.
- Anpassung von Meilensteinen muss im Arbeitsjournal erwähnt werden.

Dokumentation:

- Seitenumbruch muss mit Vorsicht verwendet werden (es besteht ein Kriterium dafür).
- Die technische Risikoanalyse darf durch die Risikoanalyse gemäss HERMES-gibb ersetzt werden.
- Es müssen möglichst viele Diagramme verwendet werden.
- Die Diagramme müssen dem Standard entsprechen.
- Es dürfen keine eigenen Einführungen für Diagramme verwendet werden.
- Die Diagramme müssen mit einer Beschreibung erklärt werden.
- Die Beschreibung muss kurz sein.
- Es dürfen grundsätzlich Fragen zum Fachbereich gestellt werden aber nur, nachdem ich selbst probiert habe.
- Die Hilfestellung muss im Arbeitsjournal dokumentiert werden.
- Die Rechtschreibung darf durch Dritte überprüft werden außer Herrn Hofmann und Herrn Baumgartner.
- Die Rechtsreibkorrekturen müssen transparent sein (daher immer eine Version vor den Korrekturen und eine Version nach den Korrekturen).

- Die Kapiteln-Liste aus dem Dokumentenvorgabe, dass Pkorg bietet, können weggelassen werden aber mit einer Begründung, auf dem Arbeitsjournal oder Sitzungsprotokoll, wieso dies gemacht wurde.
- Ich muss darauf achten, dass es keine Redundanzen im Dokument vorkommen.
- Die Wirtschaftlichkeit muss kurz beschrieben werden.
- Die Probleme, die ich, während der IPA habe und deren Lösung, müssen beschrieben werden.

Anhänge:

- Es muss ersichtlich sein welcher Teil vom Code von mir ist und welcher nicht.
- Der Code von verwendeten Bibliotheken kann weggelassen werden.
- Anhänge dürfen als separate Dokumente sein.

Allgemein:

- Der Arbeit wird in einen Level «Junior» (wie nach Lehrabschluss) erwartet.
- Nicht in den letzten 10 Minuten den Upload von Dokumenten durchführen.
- Zweite Expertenbesuch wird verschoben.

Arbeitsplatz:

- Der Arbeitsplatz ist in Ordnung.
- Das Thema Dateien Wiederherstellung wird noch einmal während des zweiten Expertenbesuchs überprüft.

Aufgaben	Verantwortlich
Eine Kopie von Protokoll an alle Abwesende per E-Mail zuschicken.	Yusup Khasbulatov
Die aktuelle und die nach dem Meeting angepasste Version des Zeitplans müssen anschliessend in die Pkorg hochgeladen werden.	Yusup Khasbulatov

Tabelle 69 Sitzungsprotokoll 2

20.2.3. Sitzung 3 Datum 05. Mai 2022

Thema	Daily-Meeting
Projekt	Wackeldisplay Wetter
Anwesende	1. Ramun Hofmann 2. Yusup Khasbulatov
Abwesende	
Sitzungsort	Eigerstrasse 12 89grad GmbH
Protokollführer	Yusup Khasbulatov
Traktanden	<ul style="list-style-type: none"> • Was wurde gemacht • Wo es Problemen liegt • Was ist geplant
Beschlüsse	
Protokoll	<ul style="list-style-type: none"> • Ab heute wird das Protokoll als einen Beschlussprotokoll durchgeführt. • Keine Abwesende für Daily-Meetings erfassen (wenn es keinen Spezialfall ist).
Kurzfassung IPA	<ul style="list-style-type: none"> • Bestehende Kurzfassung muss vereinfacht werden.
Einleitung	

- Die Beschreibung der Firma kann aus dem 89grad GmbH Webseite genommen werden.
- Die Grafiken soll ich selbst erstellen falls nötig

Dokumentation

- Nebenexperte muss angepasst werden da es nun ein neuen Nebenexperte im Pkorg erfasst wurde.
- Grenzwerte für Orte und Trend sind abgeklärt und Grenzwerte für Temperatur, Zustand und Niederschlag werden im Konzeptphase vertieft.

Aufgaben	Verantwortlich
Recherche über Wetter	Yusup Khasbulatov

Tabelle 70 Sitzungsprotokoll 3

20.2.4. Sitzung 4 Datum 06. Mai 2022

Thema	Variantenentscheid und Projekt freigabe
Projekt	Wackeldisplay Wetter
Anwesende	1. Ramun Hofmann 2. Florian Baumgartner 3. Yusup Khasbulatov
Abwesende	
Sitzungsort	Eigerstrasse 12 89grad GmbH
Protokollführer	Yusup Khasbulatov
Traktanden	<ul style="list-style-type: none"> • Besprechen die Varianten • Variantenempfehlung • Variante Auswählen • Projekt Freigeben
Beschlüsse	
Variantenwahl	<ul style="list-style-type: none"> • Das Projekt wird wie empfohlen mit das OpenWeatherApi durchgeführt
Projektpause	<ul style="list-style-type: none"> • Das Projekt ist freigegeben und kann durchgeführt werden
Grenzwerte	<ul style="list-style-type: none"> • Temperatur soll in folgende Werte aufgeteilt werden: <ul style="list-style-type: none"> ◦ Tiefer als -5 ◦ Ungefähr 0 ◦ Höher als 5 ◦ Höher als 10 ◦ Höher als 20 ◦ Höher als 25 • Zustand soll wie auf der OpenWeatherApi definiert werden: <ul style="list-style-type: none"> ◦ clear sky ◦ few clouds ◦ scattered clouds ◦ broken clouds ◦ shower rain ◦ rain

- thunderstorm
- snow
- mist
- Für den Niederschlag soll eine Recherche gemacht werden und herausfinden welche Werte sind passend
- Der Trend soll von drei Tagen berechnet werden.

Tabelle 71 Sitzungsprotokoll 4

20.2.5. Sitzung 5 Datum 09. Mai 2022

Thema	Daily Meeting
Projekt	Wackeldisplay Wetter
Anwesende	1. Ramun Hofmann 2. Yusup Khasbulatov
Abwesende	
Sitzungsort	Eigerstrasse 12 89grad GmbH
Protokollführer	Yusup Khasbulatov
Traktanden	<ul style="list-style-type: none"> ● Was wurde gemacht ● Wo es Problemen liegt ● Was ist geplant
Beschlüsse	
Zeitplan	<ul style="list-style-type: none"> ● Nach dem letzten Meilenstein soll keine Aufgaben geplant werden ● Zeitplan muss in einer PDF an den Experten und FV per E-Mail gesendet werden ● Die Notizen, die ich für die späteren Ergänzung der Dokumentation gemacht habe, könnte als Aufgaben in Zeitplan definiert werden.
Dokumentation	<ul style="list-style-type: none"> ● Darstellung von Dokumentation muss regelmässig getestet werden in dem ich es als PDF exportiere. ● ISDS Konzept auf bestehende Schutzbedarfsanalyse beziehen.

Tabelle 72 Sitzungsprotokoll 5

20.2.6. Sitzung 6 Datum 12. Mai 2022

Thema	Phasenfreigabe (Realisierung)
Projekt	Wackeldisplay Wetter
Anwesende	1. Ramun Hofmann 2. Florian Baumgartner 3. Yusup Khasbulatov
Abwesende	
Sitzungsort	Eigerstrasse 12 89grad GmbH
Protokollführer	Yusup Khasbulatov
Traktanden	<ul style="list-style-type: none"> ● Konzeptbericht durchlesen

	<ul style="list-style-type: none"> • Wireframes Besprechen • Datendarstellung besprechen
Beschlüsse	
Wireframes	
<ul style="list-style-type: none"> • Die Webseite kann wie auf der Wireframes erstellt werden 	
Datenmodell	
<ul style="list-style-type: none"> • Die Wetter Daten müssen gespeichert werden für den Fall, wenn das System offline geht oder für zukünftige Visualisierung • Die fünfte Spalte des Wackeldisplays soll drei Symbole zeigen: <ul style="list-style-type: none"> ◦ Wetter bleibt gleich: -> ◦ Wetter wird besser: ^ ◦ Wetter wird schlechter: v • Ich muss einen ERD für die Abspeicherung der Daten erstellen. • Ich muss einen Vorschlag für die Datenform von Niederschlagwerte erstellen basierend auf dem Daten des letzten Jahres. • Die Daten soll jede Stunde geholt werden und in einer Tabelle gespeichert werden • Einem Vorschlage machen, wie die Datenbank normalisiert werden soll 	
Coding-Convention	
<ul style="list-style-type: none"> • Für Code Qualitätssicherung kann ich flake8 verwenden. 	
Phasenfreigabe	
<ul style="list-style-type: none"> • Die Phase Realisierung wurde freigegeben 	

Tabelle 73 Sitzungsprotokoll 6

20.2.7. Sitzung 7 Datum 13. Mai 2022

Thema	Daily Meeting
Projekt	Wackeldisplay Wetter
Anwesende	1. Ramun Hofmann 2. Yusup Khasbulatov
Abwesende	
Sitzungsort	Eigerstrasse 12 89grad GmbH
Protokollführer	Yusup Khasbulatov
Traktanden	<ul style="list-style-type: none"> • Was wurde gemacht • Wo es Problemen liegt • Was ist geplant

Beschlüsse**Datenbank**

- Es braucht noch zusätzliche Tabelle für «Symbole»
- Tägliche Daten können einmal pro Tag angefragt werden

Tabelle 74 Sitzungsprotokoll 7

20.2.8. Sitzung 8 Datum 13. Mai 2022

Thema	Zweites Expertenbesuch
Projekt	Wackeldisplay Wetter

Anwesende	1. Hans Engler 2. Georg Achermann 3. Ramun Hofmann 4. Florian Baumgartner 5. Yusup Khasbulatov
Abwesende	
Sitzungsort	Eigerstrasse 12 89grad GmbH
Protokollführer	Yusup Khasbulatov
Traktanden	<ul style="list-style-type: none"> • Zeitplan Feedback • Projekt Struktur • Dokumentation • Arbeitsplatz • Arbeitsjournal
Beschlüsse	
Zeitplan	<ul style="list-style-type: none"> • Ich darf keine Fragen zum Zeitplan stellen, und die Experten dürfen keine Empfehlungen machen
Dokumentation	<ul style="list-style-type: none"> • Ich sollte keinen Code aus externen Bibliotheken oder Paketen als Anhang beifügen. • Wenn Code aus externen Bibliotheken oder Paketen beigelegt werden soll, sollte erkennbar sein, was von mir stammt und was nicht von mir geschrieben wurde. • Die Dokumentation muss kleiner als 50mb sein. • Korrekturen von Herrn Guggisberg sind erlaubt, solange sie der deutschen Rechtschreibung folgen. Wenn es technische Korrekturen gibt, müssen diese als Hilfsmittel in das Arbeitsjournal eingetragen werden.
Arbeitsjournal	<ul style="list-style-type: none"> • Arbeitsjournal ist gut gemacht (weiter so)
Vortrag	<ul style="list-style-type: none"> • Für die Präsentation muss ein Raum reserviert werden • Die Präsentation wird von Herrn Engler eröffnet • Ich sollte eine Einführung in das Unternehmen und das Projekt geben. • Die Präsentation sollte 17 Minuten dauern • Ich soll die Präsentation laut und deutlich vortragen • Ich soll zeigen, worauf ich stolz bin und was ich besonders gut finde. • Es muss weniger Multimedia enthalten (mehr auf technische Aspekte achten). • Ich muss zeigen, wie ich Probleme gelöst habe (Wiederholung der Dokumentation ist langweilig) • Nach der Präsentation Fachgespräch. • Es wird eine Vertiefung in 6 Themenblöcken geben • Erste Frage von 89grad GmbH (Herr Hofmann) dann Experten

Tabelle 75 Sitzungsprotokoll 8

20.2.9. Sitzung 9 Datum 16. Mai 2022

Thema	Daily Meeting
Projekt	Wackeldisplay Wetter

Anwesende	1. Ramun Hofmann 2. Yusup Khasbulatov
Abwesende	
Sitzungsort	Eigerstrasse 12 89grad GmbH
Protokollführer	Yusup Khasbulatov
Traktanden	<ul style="list-style-type: none"> • Was wurde gemacht • Wo es Problemen liegt • Was ist geplant
Beschlüsse	
Datenbank	<ul style="list-style-type: none"> • Es braucht noch zusätzliche Tabelle für «Symbole» • Tägliche Daten können einmal pro Tag angefragt werden

Tabelle 76 Sitzungsprotokoll 9

20.2.10. Sitzung 10 Datum 19. Mai 2022

Thema	Phasenfreigabe (Einführung)
Projekt	Wackeldisplay Wetter
Anwesende	1. Ramun Hofmann 2. Florian Baumgartner 3. Yusup Khasbulatov
Abwesende	
Sitzungsort	Eigerstrasse 12 89grad GmbH
Protokollführer	Yusup Khasbulatov
Traktanden	<ul style="list-style-type: none"> • Initialisierungsskripte • Workers
Beschlüsse	
Workers	<ul style="list-style-type: none"> • Die Pausenzeit zwischen den Schleifenschritten muss verkürzt werden. • Das Ausgabeformat der Daten in der Konsole muss in tabellarischer Form dargestellt werden.
Datenbank	<ul style="list-style-type: none"> • Die MAC-Adressen müssen vorher notiert werden • Die MAC-Adressen müssen in der Reihenfolge notiert werden.

Tabelle 77 Sitzungsprotokoll 10

20.2.11. Sitzung 11 Datum 23. Mai 2022

Thema	Projektabnahme
Projekt	Wackeldisplay Wetter
Anwesende	1. Ramun Hofmann 2. Florian Baumgartner 3. Yusup Khasbulatov

Abwesende	
Sitzungsort	Eigerstrasse 12 89grad GmbH
Protokollführer	Yusup Khasbulatov
Traktanden	Weiterentwicklung
Beschlüsse	
Wackeldisplay	<ul style="list-style-type: none">• Geschwindigkeit der Motoren wird als Ziel für Weiterentwicklung des Projekts definiert.• Symbole müssen vor der Präsentation draufgeklebt werden.

Tabelle 78 Sitzungsprotokoll 11

20.3. Coding Convention vom 89grad GmbH

Allgemein

20.3.1. Kommentare

Der Code wird immer in Englisch kommentiert. Keine andere Sprache ist erlaubt.

Python-Code

89grad GmbH folgt in seine Projekte nach pep8⁹ Regeln. Nachfolgend ist die Liste der Regeln, die 89grad GmbH in der internen Wikipedia zur Verfügung hat.

20.3.2. Leere Zeilen

Verwenden Sie zwei Leerzeilen vor Klassen oder Funktionen (nicht in Klassen).

Verwenden Sie eine Leerzeile vor Methoden (Funktionen von Klassen).

Eine Leerzeile kann verwendet werden, um den Code in Blöcke zu gliedern.

Verwenden Sie dafür aber nicht mehr als eine Leerzeile.

Fügen Sie immer eine Leerzeile am Ende jeder Datei ein.

20.3.3. Kommentare

Kommentare zu Funktionen und Klassen gehören in Kommentare wie diese: """Diese Klasse repräsentiert grüne Einhörner.""""", die direkt auf die Deklarationszeile folgen.

Andere Kommentare werden mit '#', gefolgt von einem Leerzeichen, kommentiert. Diese Kommentare können in der gleichen Zeile wie der Code oder in separaten Zeilen stehen.

20.3.4. Benennungskonventionen

Klassennamen sind immer in CamelCase-Schreibweise. Beispiele: GreenUnicorn, Penguin

Attributnamen sind immer Kleinbuchstaben, mehrere Wörter durch Unterstriche voneinander getrennt. Beispiele: green_unicorn, penguin, custom_penguin

Funktions- und Methodennamen (wie eine Funktion, aber als Teil einer Klasse) werden immer kleingeschrieben, mehrere Wörter durch Unterstriche voneinander getrennt: Beispiele: paint_unicorns, count_penguins, is_valid

Konstantennamen sind alle Grossbuchstaben mit Unterstrichen zwischen den Wörtern. Beispiel: MAX_PENGUINS.

Dateinamen sind immer Kleinbuchstaben, mehrere Wörter durch Unterstriche voneinander getrennt. Beispiele: admin.py, utils.py, unicorn_views.py

20.3.5. Whitespaces

Verwenden Sie ein Leerzeichen vor und nach dem =-Zeichen, wenn Sie einer Variablen etwas zuweisen wollen. Beispiel: x = 1

⁹ Siehe: <https://peps.python.org/pep-0008/>

Verwenden Sie keine Leerzeichen um das =-Zeichen, wenn Sie ein Schlüsselwortargument oder einen Standardparameterwert angeben. Beispiele: add(a=1,b=3), def add(a=1,b=1)

20.4. Design Guidelines vom 89grad GmbH

Farbpalette



Abbildung 40 Design Guidelines: Farbpalette

Farbe	HEX	RGB	HSL
	#8BC53F	rgb(139, 197, 63)	hsl(85.97, 53.6%, 50.98%)
	#5E5E5E	rgb(94, 94, 94)	hsl(0, 0%, 36.86%)
	#3E313C	rgb(62, 49, 60)	hsl(309.23, 11.71%, 21.76%)
	#FFFFFF	rgb(255, 255, 255)	hsl(0, 0%, 100%)
	#000000	rgb(0, 0, 0)	hsl(0, 0%, 0%)

Tabelle 79 Design Guidelines: Farbpalette

Farbtöne



Standard: #8BC53F	Dunkel: #599400	Standard: #5E5E5E	Dunkel: #343434	Standard: #3E313C	Dunkel: #180916	Standard: #FFFFFF	Dunkel: #CCCCCC	Standard: #000000	Dunkel: #000000

Tabelle 80 Design Guidelines: Farbtöne

20.5. Testprotokoll

Testname	QR-Code wird als einen Link erkannt.	
Testnummer	T-01	
Testfall Nummer	FT1	
Testperson	Abdelrahman Moustafa	
Testzeitpunkt	20.05.2022 15:00	
Testvoraussetzungen		
Testschritte	<ol style="list-style-type: none"> 1. Der Tester/in muss sich mit dem Wi-Fi namens «89grad» oder «89guest» verbinden. 2. Der Tester/in scannt den QR-Code auf dem Wackeldisplay 	
Erwartete Ergebnis	Der QR-Code wird als Link erkannt.	
Effektives Ergebnis	Der QR-Code wird als Link erkannt.	Erfüllt?
		Ja

Tabelle 81 Testprotokoll: T-01

Testname	Wackeldisplay startet	
Testnummer	T-02	
Testfall Nummer	FT2	
Testperson	Abdelrahman Moustafa <i>A. moustafa</i>	
Testzeitpunkt	20.05.2022 15:05	
Testvoraussetzungen	T-01	
Testschritte	1. Der Tester/in öffnet die URL (auf manchen Handy-Versionen kann es sein, dass die URL automatisch geöffnet wird)	
Erwartete Ergebnis	Das Wackeldisplay startet innerhalb von 1-10 Sekunden. Nach 3 Minuten fahren alle Spalten wieder auf die Initiale Position.	
Effektives Ergebnis	<p>Q-Bemerkung: URL wurde nicht automatic geöffnet.</p> <ul style="list-style-type: none"> - Das Wackeldisplay startet innerhalb von ~1-10 S. - Nach (3 bis 4) Minuten haben die Spalten nach unten gefallen 	Erfüllt?
	<p>Ja (teilweise)</p>	

Tabelle 82 Testprotokoll: T-02

Testname	Daten werden auf der Website angezeigt	
Testnummer	T-03	
Testfall Nummer	FT3	
Testperson	Abdelrahman Moustafa <i>A. moustafa</i>	
Testzeitpunkt	20.05.2022 15:10	
Testvoraussetzungen	T-02	
Testschritte	1. Der Tester/in stellt sicher, dass die Daten auf der Webseite angezeigt werden.	
Erwartete Ergebnis	Die Daten werden auf der Webseite in eine Tabelle dargestellt.	
Effektives Ergebnis	<p>Die Daten werden auf der website gezeigt. 6 Spalten. 1-ort, 2-tempatur, 3-zustand, 4-niederschlag 5-trend, 6-Aktion</p>	Erfüllt?
	<p>Ja</p>	

Tabelle 83 Testprotokoll: T03

Testname	Bestimmte Daten werden auf dem Wackeldisplay angezeigt
Testnummer	T-04
Testfall Nummer	FT4
Testperson	Abdelrahman Moustafa <i>A. Moustafa</i>
Testzeitpunkt	20.05.2022 15:15
Testvoraussetzungen	T-03
Testschritte	<ol style="list-style-type: none"> Der Tester/in klickt auf den Button «Zeigen» in der gewünschten Zeile. Der Tester/in klickt auf den Button «Bestätigen», der sich in dem erscheinenden Modal (neues Fenster) befindet.
Erwartete Ergebnis	Das Wackeldisplay startet innerhalb von 1-10 Sekunden. Ausgewählte Daten werden auf dem Wackeldisplay angezeigt. Nach 3 Minuten fahren alle Spalten wieder auf die Initiale Position.
Effektives Ergebnis	<p>Auf "Bern" Zeile zeigen gedrückt. Alle Spalten haben sich bewegt und die richtige daten angezeigt und bewegt.</p> <p><i>Ja</i></p>
	Erfüllt?

Tabelle 84 Testprotokoll: T-04

Testname	Bestimmte Daten werden auf der Wackeldisplay nicht angezeigt
Testnummer	T-05
Testfall Nummer	FT5
Testperson	Abdelrahman Moustafa <i>A. Moustafa</i>
Testzeitpunkt	20.05.2022 15:20
Testvoraussetzungen	T-02
Testschritte	<ol style="list-style-type: none"> Der Tester/in klickt auf den Button «Zeigen» in der gewünschten Zeile.
Erwartete Ergebnis	Es erscheint ein Modal (neues Fenster), in dem man sieht, dass die Wackelanzeige nicht gestartet wird, weil sie bereits in Betrieb ist.
Effektives Ergebnis	<p>Auf "Bern" (innerhalb von 3-bis 4 minuten) nochmal gedrückt. ein modal mit meldung ist gekommen "wackel display in Betrieb, versuchen sie es später noch mal".</p> <p>- Spalten haben sich nicht bewegt "No action take place")</p> <p><i>Ja</i></p>
Zeigen btn	

Tabelle 85 Testprotokoll: T-05

20.6. Dokumentvorlage 89grad GmbH

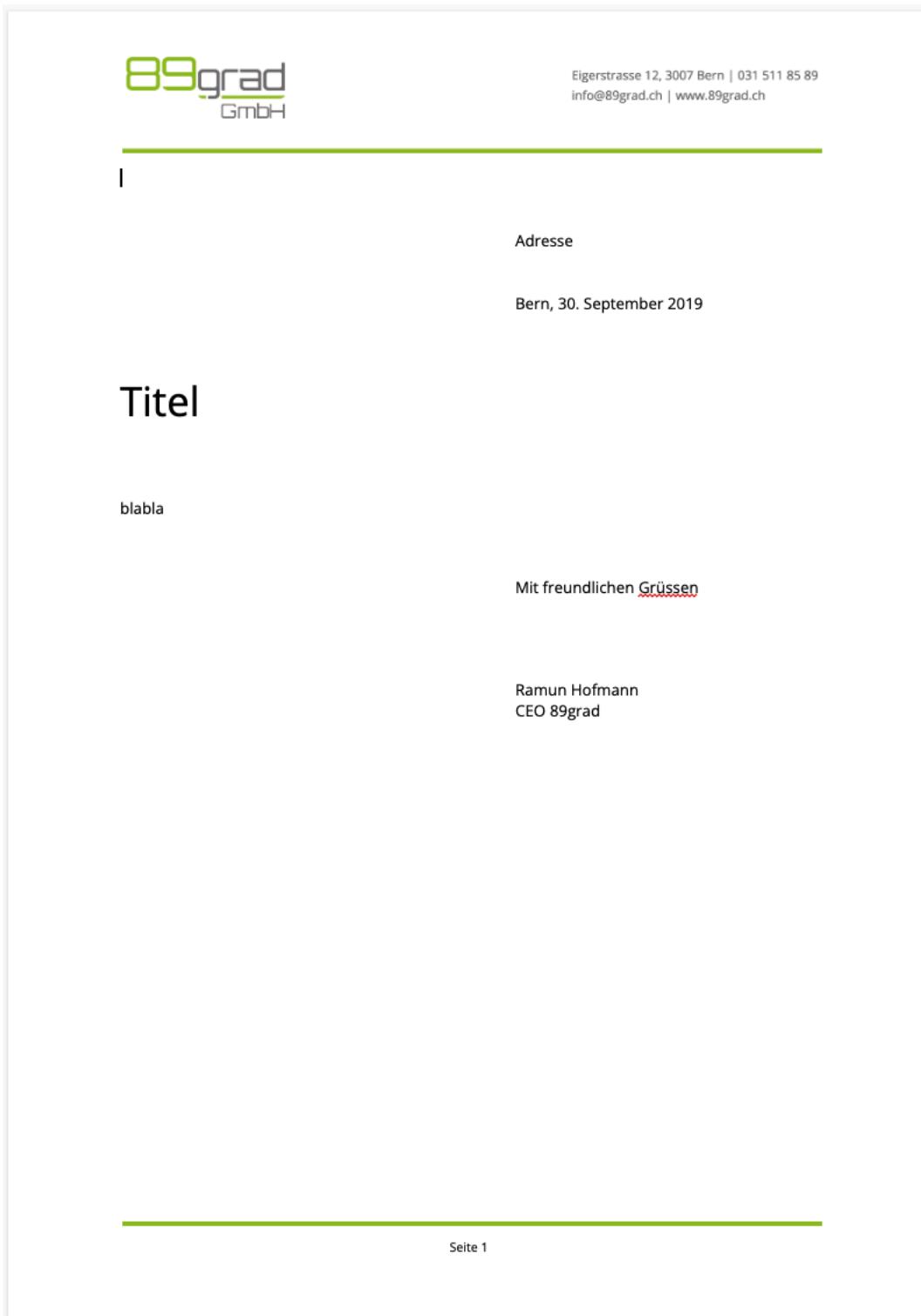
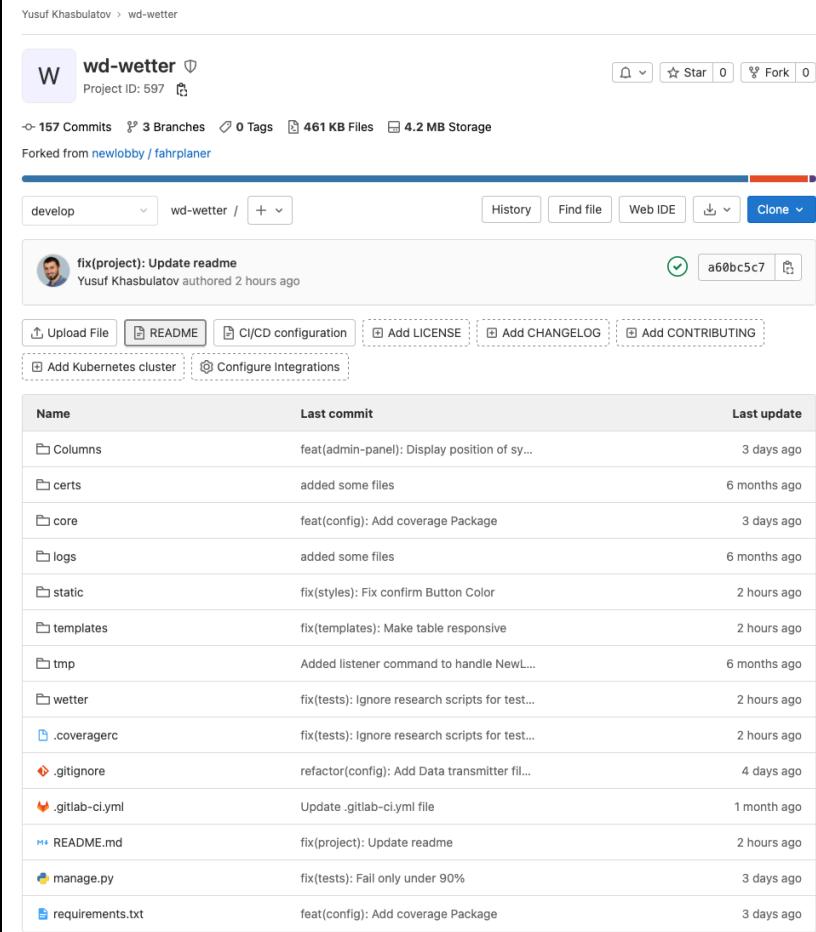


Abbildung 41 Anhang: 89grad GmbH Dokumentenvorlage

20.7. Versionsverwaltung mit Verwaltungssoftware

Wie in den Standards beschrieben, verwendet die 89grad GmbH einen eigenen GitLab-Server für die Versionierung. Das Projekt wird auch auf dieser Plattform versioniert.

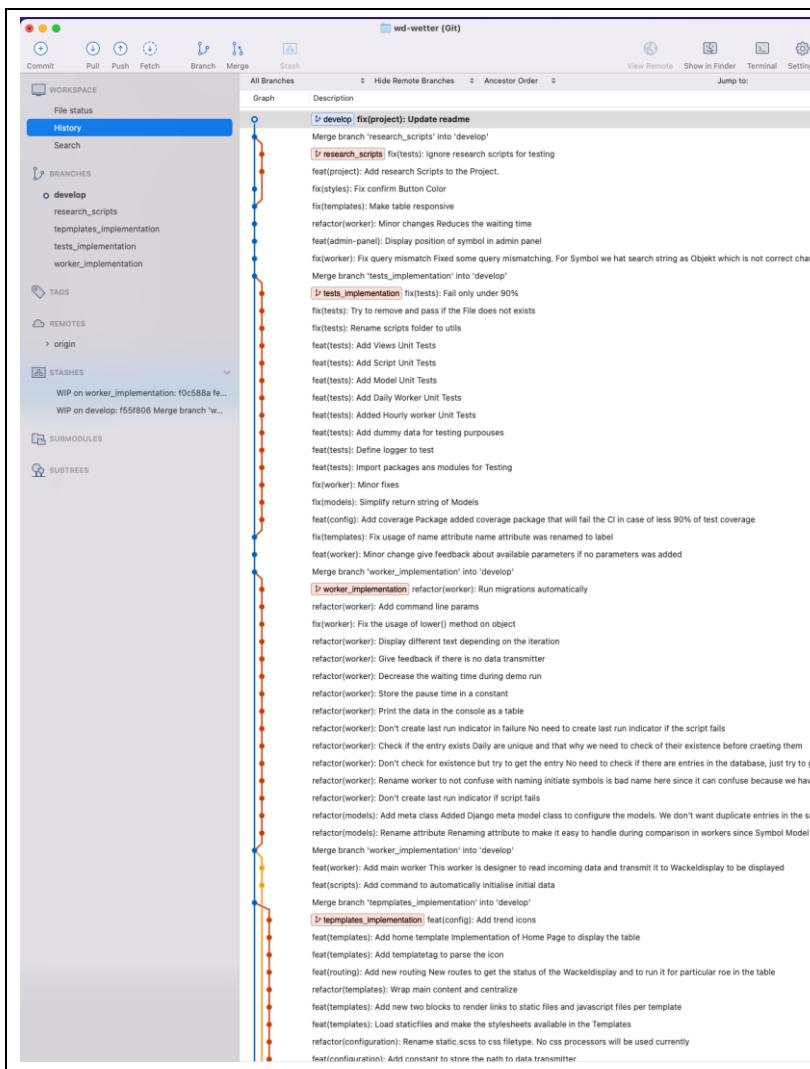
Screenshot	Beschreibung																																													
 <p>The screenshot shows a GitLab project page for 'wd-wetter'. At the top, it displays basic project statistics: 157 Commits, 3 Branches, 0 Tags, 461 KB Files, and 4.2 MB Storage. The project was forked from 'newlobby / fahrplaner'. Below this, there's a navigation bar with dropdowns for 'develop' and 'wd-wetter / +', and buttons for 'History', 'Find file', 'Web IDE', 'Clone', and a download icon. A prominent commit message from Yusuf Khasbulatov is shown: 'fix(project): Update readme' (commit a60bc5c7). Below the commit message, there are several dashed buttons for actions like 'Upload File', 'README', 'CI/CD configuration', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', 'Add Kubernetes cluster', and 'Configure Integrations'. The main area of the screenshot is a table showing the commit history:</p> <table border="1"><thead><tr><th>Name</th><th>Last commit</th><th>Last update</th></tr></thead><tbody><tr><td>Columns</td><td>feat(admin-panel): Display position of sy...</td><td>3 days ago</td></tr><tr><td>certs</td><td>added some files</td><td>6 months ago</td></tr><tr><td>core</td><td>feat(config): Add coverage Package</td><td>3 days ago</td></tr><tr><td>logs</td><td>added some files</td><td>6 months ago</td></tr><tr><td>static</td><td>fix(styles): Fix confirm Button Color</td><td>2 hours ago</td></tr><tr><td>templates</td><td>fix(templates): Make table responsive</td><td>2 hours ago</td></tr><tr><td>tmp</td><td>Added listener command to handle NewL...</td><td>6 months ago</td></tr><tr><td>wetter</td><td>fix(tests): Ignore research scripts for test...</td><td>2 hours ago</td></tr><tr><td>.coveragerc</td><td>fix(tests): Ignore research scripts for test...</td><td>2 hours ago</td></tr><tr><td>.gitignore</td><td>refactor(config): Add Data transmitter fil...</td><td>4 days ago</td></tr><tr><td>.gitlab-ci.yml</td><td>Update .gitlab-ci.yml file</td><td>1 month ago</td></tr><tr><td>README.md</td><td>fix(project): Update readme</td><td>2 hours ago</td></tr><tr><td>manage.py</td><td>fix(tests): Fail only under 90%</td><td>3 days ago</td></tr><tr><td>requirements.txt</td><td>feat(config): Add coverage Package</td><td>3 days ago</td></tr></tbody></table>	Name	Last commit	Last update	Columns	feat(admin-panel): Display position of sy...	3 days ago	certs	added some files	6 months ago	core	feat(config): Add coverage Package	3 days ago	logs	added some files	6 months ago	static	fix(styles): Fix confirm Button Color	2 hours ago	templates	fix(templates): Make table responsive	2 hours ago	tmp	Added listener command to handle NewL...	6 months ago	wetter	fix(tests): Ignore research scripts for test...	2 hours ago	.coveragerc	fix(tests): Ignore research scripts for test...	2 hours ago	.gitignore	refactor(config): Add Data transmitter fil...	4 days ago	.gitlab-ci.yml	Update .gitlab-ci.yml file	1 month ago	README.md	fix(project): Update readme	2 hours ago	manage.py	fix(tests): Fail only under 90%	3 days ago	requirements.txt	feat(config): Add coverage Package	3 days ago	<p>Obere Übersicht Übersicht über die Projekte im GitLab.</p>
Name	Last commit	Last update																																												
Columns	feat(admin-panel): Display position of sy...	3 days ago																																												
certs	added some files	6 months ago																																												
core	feat(config): Add coverage Package	3 days ago																																												
logs	added some files	6 months ago																																												
static	fix(styles): Fix confirm Button Color	2 hours ago																																												
templates	fix(templates): Make table responsive	2 hours ago																																												
tmp	Added listener command to handle NewL...	6 months ago																																												
wetter	fix(tests): Ignore research scripts for test...	2 hours ago																																												
.coveragerc	fix(tests): Ignore research scripts for test...	2 hours ago																																												
.gitignore	refactor(config): Add Data transmitter fil...	4 days ago																																												
.gitlab-ci.yml	Update .gitlab-ci.yml file	1 month ago																																												
README.md	fix(project): Update readme	2 hours ago																																												
manage.py	fix(tests): Fail only under 90%	3 days ago																																												
requirements.txt	feat(config): Add coverage Package	3 days ago																																												

The screenshot shows a list of commits from the 'wd-wetter' project on GitLab. The commits are grouped by date:

- 23 May, 2022**: 6 commits
 - fix(project): Update readme
 - Merge branch 'research_scripts' into 'develop'
 - fix(styles): Fix confirm Button Color
 - fix(tests): Ignore research scripts for testing
 - feat(project): Add research Scripts to the Project.
 - fix(templates): Make table responsive
- 20 May, 2022**: 3 commits
 - refactor(worker): Minor changes
 - feat(admin-panel): Display position of symbol in admin panel
 - fix(worker): Fix query mismatch
- 19 May, 2022**: 31 commits
 - Merge branch 'tests_implementation' into 'develop'
 - fix(tests): Fail only under 90%
 - fix(tests): Try to remove and pass if the File does not exists
 - fix(tests): Rename scripts folder to utils
 - feat(tests): Add Views Unit Tests
 - feat(tests): Add Script Unit Tests
 - feat(tests): Add Model Unit Tests
 - feat(tests): Add Daily Worker Unit Tests
 - feat(tests): Added Hourly worker Unit Tests
 - feat(tests): Add dummy data for testing purposes
 - feat(tests): Define logger to test
 - feat(tests): Import packages ans modules for Testing
 - fix(worker): Minor fixes
 - fix(models): Simplify return string of Models
 - feat(config): Add coverage Package

Commits

Übersicht über die letzte Commits im GitLab.



Branches

Übersicht über die Branches.

Der Screenshot wurde in der Sourcetree-Anwendung erstellt, da die Branchen auf der Remote bereits gelöscht wurden (automatisch nach dem Mergen).

The screenshot shows the GitLab CI Pipelines interface. At the top, there are filters for 'All' (29), 'Finished', 'Branches', and 'Tags'. Below the filters is a search bar and buttons for 'Clear runner caches', 'CI lint', and 'Run pipeline'. A 'Filter pipelines' input field and a 'Show Pipeline ID' dropdown are also present. The main area displays a table of pipelines:

Status	Pipeline	Triggerer	Stages
passed	fix(project): Update readme #14803 ➔ develop -> a60bc5c7 [test]	2 hours ago	Passed
passed	Merge branch 'research_scripts' into 'develop' #14802 ➔ develop -> 3963a683 [test]	2 hours ago	Passed
passed	fix(styles): Fix confirm Button Color #14801 ➔ develop -> 5f7b0a81 [test]	2 hours ago	Passed
passed	fix(tests): Ignore research scripts for testing #14800 ➔ research_scripts -> 5ce7db84 [test]	2 hours ago	Passed
failed	feat(project): Add research Scripts to the Project #14799 ➔ research_scripts -> 2684be25 [test]	2 hours ago	Passed, Failed
passed	fix(templates): Make table responsive #14798 ➔ develop -> 23382f11 [test]	2 hours ago	Passed
passed	refactor(worker): Minor changes #14787 ➔ develop -> 63528f66 [test]	3 days ago	Passed
passed	feat(admin-panel): Display position of symbol in table #14786 ➔ develop -> 4b4de379 [test]	3 days ago	Passed
passed	Merge branch 'tests_implementation' into 'dev' #14781 ➔ develop -> 53dbb055 [test]	3 days ago	Passed
passed	fix(tests): Fail only under 90% #14780 ➔ tests_implementation -> 88a925a7 [test]	3 days ago	Passed
failed	fix(tests): Try to remove and pass if the file does not exist #14779 ➔ tests_implementation -> 6414f8f6 [test]	3 days ago	Passed, Failed
failed	fix(tests): Rename scripts folder to utils #14778 ➔ tests_implementation -> f4238ab4 [test]	3 days ago	Passed, Failed
failed	feat(tests): Add Views Unit Tests #14777 ➔ tests_implementation -> d98ad469 [test]	3 days ago	Passed, Failed
passed	fix(templates): Fix usage of name attribute #14773 ➔ develop -> e48ac7e0 [test]	4 days ago	Passed
passed	feat(worker): Minor change #14772 ➔ develop -> 0a5b8115 [test]	4 days ago	Passed

Pipelines (GitLab CI)

Übersicht über die CI Jobs.

Tabelle 86 Anhang: Software-Versionsverwaltung

20.8. Ordnerstruktur und Ablage der Dokumentationen

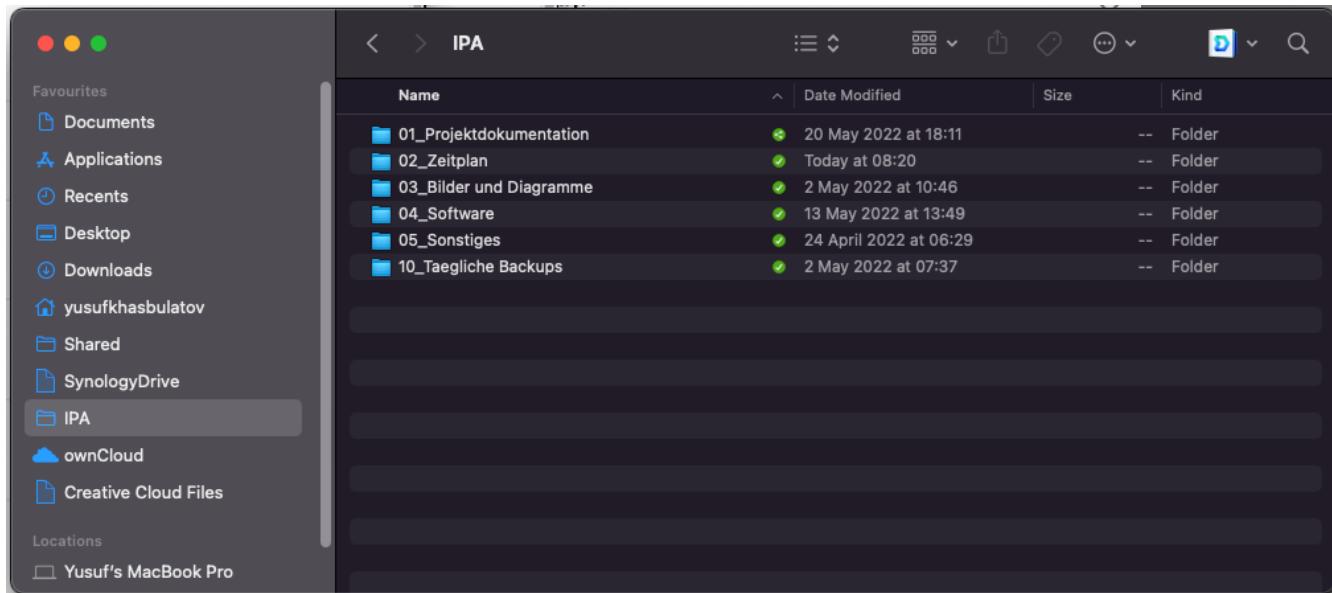


Abbildung 42 Anhang: Dokumentenablage und Ordnerstruktur

20.9. Projekt Code

Die mit einem Dollarzeichen (\$) gekennzeichneten Zeilen sind von mir geschrieben (Das gilt nur für die zweispaltige Tabelle, den Quellcode in den einspaltigen Tabellen habe ich selbst geschrieben).

Folgende Verzeichnisse und Dateien sind nicht von mir erstellt und müssen nicht berücksichtigt werden:

```
Columns/
certs/
core/__init__.py
core/asgi.py
core/settings.py
core/wsgi.py
logs/
wetter/apps.py
```

Tabelle 87 Dateien, die zu ignorieren sind

20.9.1. README Datei

```
# Deployment
ssh into Wackeldisplay server. The hostname is fahrplaner and was not changed.
```
ssh -A fahrplaner@fahrplaner.e12
```
CD into wd-wetter directory and pull the latest changes
```
cd wd-wetter
git pull
```
Check if there are active worker under screen sessions:
```
screen -ls
```
If there are no active screen sessions, start a new one:
```
screen -S wd-wetter
```
Else switch to the screen session:
```
screen -r wd-wetter
```
Activate the virtual environment:
```
workon wetter
```
Run the server:
```
touch core/wsgi.py
```
Run the worker:
```

```

```
./manage.py wetter_worker
```
```

Tabelle 88 Anhang: Readme - README.md

20.9.2. Wackeldisplay Worker (Python Skripte)

Es wurden zwei Worker entwickelt, einer für den stündlichen und einer für den täglichen Abruf von Daten.

Datei wetter/management/commands/pull_hourly.py



```
import json
import logging
from datetime import datetime, timedelta
from django.utils.timezone import make_aware
from django.core.management import BaseCommand

from core.settings_devel import LOG_DIR
from wetter.management.commands.pull_daily import create_daily_data
from wetter.models import WeatherHourly, WeatherDaily, Location

# Pulling Weather Data will be logged separately
from wetter.utils.ow_api_service import get_ow_data, get_weather_symbol_by_identifier

logger = logging.getLogger('pull_weather')

class Command(BaseCommand):
    help = 'Pull hourly data from the server'

    def handle(self, *args, **options): # pragma: no cover
        logger.info('Pull hourly data from the server')

        # First check if the last pull was made less than an hour ago
        if is_less_than_one_hour():
            return

        # Get the latest hourly data
        locations = Location.objects.all()
        for location in locations:
            data = get_ow_data(location)

            # Delegate daily data creation to another function
            create_daily_data(data, location)

            # Create hourly data
            current_data = data['current']
            weather_symbol = get_weather_symbol_by_identifier(current_data['weather'][0]['id'])
            date = make_aware(datetime.fromtimestamp(current_data['dt']))
            WeatherHourly.objects.create(
                date=date,
                temp=current_data['temp'],
```

```
        rain=current_data['rain']['1h'] if 'rain' in current_data else
0,
        trend=get_weather_trend(
            date=date,
            current_symbol=weather_symbol,
            location=location,
        ),
        weather_symbol=weather_symbol,
        location=location,
    )

# Update last_call_time.json
logger.info('Pulling hourly data finished')
logger.info('Saving datetime in last_call_time.json')
with open(LOG_DIR / 'last_call_time.json', 'w') as f:
    json.dump(datetime.now().strftime('%Y-%m-%d %H:%M:%S'), f)

def is_less_than_one_hour():
    # Open last_call_time.json file
    try:
        with open(LOG_DIR / 'last_call_time.json', 'r') as file:
            last_call_time = json.load(file)
            last_call_time = datetime.strptime(last_call_time, '%Y-%m-%d
%H:%M:%S')

        # If last call was made less than an hour ago, skip
        if (datetime.now() - last_call_time) < timedelta(hours=1):
            logger.info('Last call was made less than an hour ago. Skip-
ping.')
            return True
        else:
            return False

    except Exception as e:
        logger.info('Creating empty last_call_time.json')
        with open(LOG_DIR / 'last_call_time.json', 'w') as f:
            json.dump("{}", f)
        # If there is any error, assume that the file is empty or does not ex-
ist
        # Inform the user and continue
        logger.critical(f'error parsing json file')
        logger.critical('last_call_time.json file is corrupted. Continuing.')
        return False

def get_weather_trend(date, current_symbol, location):
    # Get WeatherDaily for next 3 days
    weather_daily = WeatherDaily.objects.filter(
        location=location,
        date__gte=date - timedelta(days=1),
        date__lte=date + timedelta(days=3),
    )
    if not weather_daily.count():
        return None

    # Calculate the average of the weather symbols in the next 3 days
    symbol_count = 0
```

```
symbol_sum = 0
for daily_weather in weather_daily:
    symbol_sum += daily_weather.symbol.rating
    symbol_count += 1

average_symbol = symbol_sum / symbol_count
if average_symbol > current_symbol.rating:
    return 1 # Trend rising
elif average_symbol < current_symbol.rating:
    return 2 # Trend falling
else:
    return 0 # Trend stable
```

Tabelle 89 Anhang: Skripte - pull_hourly.py

Datei wetter/management/commands/pull_daily.py

```
import logging
from datetime import datetime, timedelta

from django.core.management import BaseCommand
from django.db import IntegrityError
from django.utils import timezone

from wetter.models import WeatherDaily, Location
from wetter.utils.ow_api_service import get_weather_symbol_by_identifier,
get_ow_data

logger = logging.getLogger('pull_weather')

class Command(BaseCommand):
    help = 'Pull daily data from the server'

    def handle(self, *args, **options): # pragma: no cover
        logger.info('Pull daily data from the server')

        # Get the latest hourly data
        locations = Location.objects.all()
        for location in locations:
            data = get_ow_data(location)

            # Delegate daily data creation to another function
            create_daily_data(data, location)

    def create_daily_data(data, location):
        # Slice the data depending on the amount of currently available data in the
        # database
        # If there is no data in the database, then slice next 3 days
        daily_data_count = WeatherDaily.objects.filter(location=location).count()
        if daily_data_count < 3:
            data = data['daily'][1:4]
        # If there is data in the database, then just add +1 day
```

```

else:
    last_entry = WeatherDaily.objects.filter(location=location).first()
    # To avoid duplication, data is added only if
    # there is more than 72 hours difference between the last entry and today.
    if (last_entry.date - timezone.now()) > timedelta(hours=72):
        return
    data = data['daily'][3:4] # keep it as a list to loop over it later

    # Create daily data
    for day in data:
        # Convert timestamp to datetime
        date = timezone.make_aware(datetime.fromtimestamp(day['dt']))
        weather_symbol = get_weather_symbol_by_identifier(day['weather'][0]['id'])

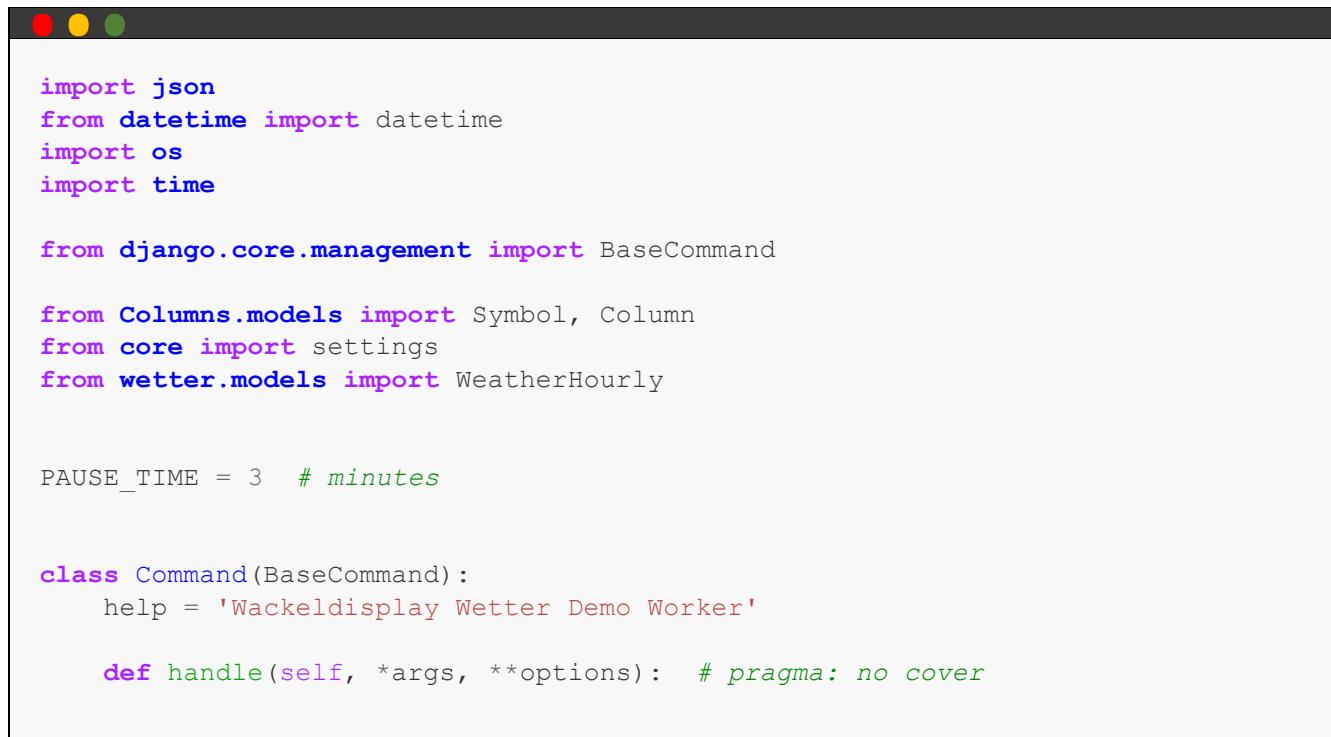
        # Create WeatherDaily object
        try:
            WeatherDaily.objects.create(
                location=location,
                date=date,
                symbol=weather_symbol,
            )
        except IntegrityError as e:
            # If the object already exists, then do nothing
            logger.info(f'Daily data already exists for {location.name} on {date}')
            pass

```

Tabelle 90 Anhang: Skripte - pull_daily.py

Wackeldisplay Steuerung übernimmt folgendes Worker Skript

Datei wetter/management/commands/wetter_worker.py



```

import json
from datetime import datetime
import os
import time

from django.core.management import BaseCommand

from Columns.models import Symbol, Column
from core import settings
from wetter.models import WeatherHourly

PAUSE_TIME = 3 # minutes

class Command(BaseCommand):
    help = 'Wackeldisplay Wetter Demo Worker'

    def handle(self, *args, **options): # pragma: no cover

```

```
while True:
    if os.path.isfile(settings.MQTT_OCCUPIED):
        file_last_modified = os.path.getmtime(os.path.join(settings.MQTT_OCCUPIED))
        self.stdout.write(
            f'File last modified:\n{datetime.fromtimestamp(file_last_modified).strftime("%Y-%m-%d %H:%M:%S") }')
        # If the file is older than PAUSE_TIME minutes
        # we can assume that the Wackeldisplay is not running anymore
        # And start another demo
        if (file_last_modified + (60 * PAUSE_TIME)) < int(round(time.time())):
            self.stdout.write(f'File is older than {PAUSE_TIME} minutes. Removing demo_started.json')
            os.remove(os.path.join(settings.MQTT_OCCUPIED))

    # If the demo_started.json file does not exist, wait 3 seconds and
    # try again
    if not os.path.isfile(settings.MQTT_OCCUPIED):
        try:
            _time = datetime.now().strftime('%H:%M:%S') # Get current
            time
            self.stdout.write(f'[{_time}] Waiting for
demo_started.json')
            time.sleep(3) # 3 seconds
            continue
        except KeyboardInterrupt:
            self.stdout.write('Stopping Listener Process')
            break

    # Open current_data.json and start the demo
    try:
        with open(settings.MQTT_CURRENT_DATA, 'r') as f:
            data = json.load(f)
    except FileNotFoundError:
        self.stdout.write('File current_data.json not found. No data
available')
        continue

    if 'show_data' in data:
        # Get the weather hourly data
        weather_hourly = WeatherHourly.objects.filter(id__in=data['show_data'])
        self.stdout.write(f'Starting the process of displaying the
data: {len(weather_hourly)}')
        self.stdout.write(self.style.SUCCESS('Demo is started'))
        for i, w in enumerate(weather_hourly):
            print_displaying_data(self.stdout, w)

            call_location_show_method(w.location, self.stdout,
self.style)
            call_temperature_show_method(w.temp, self.stdout,
self.style)
            call_weather_symbol_show_method(w.weather_symbol.label,
self.stdout, self.style)
            call_rain_show_method(w.rain, self.stdout, self.style)
            call_trend_show_method(w.trend, self.stdout, self.style)
```

```
# Wait for 30 seconds and continue
# This is needed to make sure that the symbol has enough
time to be displayed
try:
    if i < len(weather_hourly) - 1:
        self.stdout.write(f'Waiting for 30 seconds to dis-
play next symbol')
    else:
        self.stdout.write(f'Displaying the last symbol.
Waiting for 30 seconds.')
    time.sleep(30)
    continue
except KeyboardInterrupt:
    self.stdout.write('Stopping Listener Process')
# If the data not in the current_data.json file (this should not
happen) then just continue
else:
    self.stdout.write('No data available')
    continue

try:
    self.stdout.write(self.style.SUCCESS(f'Waiting for 30 seconds
before calibration starts'))
    time.sleep(30)
    self.stdout.write(self.style.SUCCESS('Demo is ended'))
    self.stdout.write(self.style.SUCCESS('Starting Display Columns
calibration'))
    for column in Column.objects.all():
        try:
            # This will drive all the columns to the initial posi-
            position
            column.calibrate()
        except Exception as e:
            self.stdout.write(self.style.ERROR(f'Error calibrating
column <{column.label}>: {e}'))
            continue
        self.stdout.write(self.style.SUCCESS('Waiting 1 minutes until
the calibration is completed'))
        time.sleep(60) # 1 minute
        self.stdout.write(self.style.SUCCESS('Removing
demo_started.json'))
        os.remove(os.path.join(settings.MQTT_OCCUPIED))
        self.stdout.write(self.style.SUCCESS('Demo is ready for next
run'))
        continue
    except KeyboardInterrupt:
        self.stdout.write(self.style.WARNING('Stopping Listener Pro-
cess'))
        break

def call_location_show_method(location, stdout, style):
    try:
        symbol = Symbol.objects.get(label=location.name.lower())
        symbol.show()
        stdout.write(style.SUCCESS(f'[{symbol.label}] {location}'))
    except Exception as e:
        stdout.write(style.ERROR(f'Error displaying the location: {e}'))
```

```
def call_temperature_show_method(temp, stdout, style):
    try:
        label = '< -5'
        if (-5 < temp <= 0) or (0 < temp <= 5):
            label = '~ 0'
        elif 5 < temp <= 10:
            label = '> 5'
        elif 10 < temp <= 20:
            label = '> 10'
        elif 20 < temp <= 25:
            label = '> 20'
        elif temp > 25:
            label = '> 25'
        symbol = Symbol.objects.get(label=label)
        symbol.show()
        stdout.write(style.SUCCESS(f'[{symbol.label}] {temp}'))
    except Exception as e:
        stdout.write(style.ERROR(f'Error displaying the temperature: {e}'))

def call_weather_symbol_show_method(weather_symbol, stdout, style):
    try:
        symbol = Symbol.objects.get(label=weather_symbol)
        symbol.show()
        stdout.write(style.SUCCESS(f'[{symbol.label}] {weather_symbol}'))
    except Exception as e:
        stdout.write(style.ERROR(f'Error displaying the weather symbol: {e}'))

def call_rain_show_method(rain, stdout, style):
    try:
        label = '0'
        if 0 < rain <= 0.1:
            label = '< 0.1'
        elif 0.1 < rain <= 1:
            label = '< 1'
        elif 1 < rain <= 2:
            label = '< 2'
        elif 2 < rain <= 3:
            label = '< 3'
        elif rain >= 3:
            label = '>= 3'
        symbol = Symbol.objects.get(label=label)
        symbol.show()
        stdout.write(style.SUCCESS(f'[{symbol.label}] {rain}'))
    except Exception as e:
        stdout.write(style.ERROR(f'Error displaying the rain: {e}'))

def call_trend_show_method(trend, stdout, style):
    try:
        label = 'equal'
        if trend == 1:
            label = 'rising'
        elif trend == 2:
            label = 'falling'
```

```
symbol = Symbol.objects.get(label=label)
symbol.show()
stdout.write(style.SUCCESS(f'[{symbol.label}] {trend}'))
except Exception as e:
    stdout.write(style.ERROR(f'Error displaying the trend: {e}'))

def print_displaying_data(stdout, data):
    stdout.write('-' * 60 + '\n')
    stdout.write(f'| Location\t| {data.location}\n')
    stdout.write(f'| Temperature\t| {data.temp}\n')
    stdout.write(f'| Symbol\t| {data.weather_symbol.label}\n')
    stdout.write(f'| Rain\t| {data.rain}\n')
    stdout.write(f'| Trend\t| {data.trend}\n')
    stdout.write('-' * 60 + '\n')
```

Tabelle 91 Anhang: Skripte - wetter_worker.py

20.9.3. Automatisation und Migration Skripte

Datei wetter/management/commands/initialize.py



```
from django.core.management.base import BaseCommand
from django.core.management import call_command
from django.db import OperationalError

from Columns.models import Column, Symbol

COLUMN_LABELS = [
    'location',
    'temperature',
    'weather',
    'rain',
    'trend'
]

SYMBOL_LABELS = {
    ######
    'location': [
        {'label': 'bern',
         'position': 1,
        },
        {'label': 'zurich',
         'position': 2,
        },
        {'label': 'paris',
         'position': 3,
        },
        {'label': 'london',
         'position': 4,
        },
        {'label': 'san francisco',
         'position': 5,
        }
    ],
    #####
}
```

```
'temperature': [{  
    'label': '> 25',  
    'position': 1,  
}, {  
    'label': '> 20',  
    'position': 2,  
}, {  
    'label': '> 10',  
    'position': 3,  
}, {  
    'label': '> 5',  
    'position': 4,  
}, {  
    'label': '~ 0',  
    'position': 5,  
}, {  
    'label': '< -5',  
    'position': 6,  
}],  
#####  
'weather': [{  
    'label': 'clear sky',  
    'position': 1,  
}, {  
    'label': 'few clouds',  
    'position': 2,  
}, {  
    'label': 'mist',  
    'position': 3,  
}, {  
    'label': 'scattered clouds',  
    'position': 4,  
}, {  
    'label': 'broken clouds',  
    'position': 5,  
}, {  
    'label': 'rain',  
    'position': 6,  
}, {  
    'label': 'shower rain',  
    'position': 7,  
}, {  
    'label': 'thunderstorm',  
    'position': 8,  
}, {  
    'label': 'snow',  
    'position': 9,  
}],  
#####  
'rain': [{  
    'label': "0",  
    'position': 1,  
}, {  
    'label': "< 0.1",  
    'position': 2,  
}, {  
    'label': "< 1",  
    'position': 3,
```

```
    },
    {
        'label': '< 2',
        'position': 4,
    },
    {
        'label': '< 3',
        'position': 5,
    },
    {
        'label': '>= 3',
        'position': 6,
    },
],
#####
'trend': [
    {
        'label': 'equal',
        'position': 1,
    },
    {
        'label': 'rising',
        'position': 2,
    },
    {
        'label': 'falling',
        'position': 3,
    },
],
}

class Command(BaseCommand):
    help = 'Initialize Columns and Symbols.'

    def add_arguments(self, parser):
        parser.add_argument('-c', '--columns', action='store_true', help='Initialize Columns.')
        parser.add_argument('-s', '--symbols', action='store_true', help='Initialize Symbols.')
        parser.add_argument('-w', '--weather', action='store_true', help='Initialize weather data.')
        parser.add_argument('-a', '--all', action='store_true', help='Initialize all.')

    def handle(self, *args, **options): # pragma: no cover
        if options['all']:
            self.stdout.write('Initializing Database...')
            self.initialize_columns(self.stdout)
            self.initialize_symbols(self.stdout)
            self.initialize_data(self.stdout)
        elif options['columns']:
            self.initialize_columns(self.stdout)
        elif options['symbols']:
            self.initialize_symbols(self.stdout)
        elif options['weather']:
            self.initialize_data(self.stdout)

        else:
            self.stdout.write('No option selected.\n')
            self.stdout.write('Following options are available:\n')
            self.stdout.write('-c, --columns\tInitialize Columns.\n')
            self.stdout.write('-s, --symbols\tInitialize Symbols.\n')
            self.stdout.write('-w, --weather\tInitialize weather data.\n')
            self.stdout.write('-a, --all\tInitialize all.\n')
            self.stdout.write('Or use -h or --help for more information.\n')


```

```
@staticmethod
def initialize_columns(stdout):
    stdout.write('Filling in the "Column" table...')
    try:
        for label in COLUMN_LABELS:
            column, created = Column.objects.get_or_create(label=label)
            if created:
                stdout.write(f'\tCreated Column: "{label}"')
            else:
                stdout.write(f'\tColumn: "{label}" already exists')
    # If the database does not exist, an OperationalError is thrown
    # We want to run migrations automatically
    except (OperationalError, Exception) as e:
        if str(e) == 'no such table: Columns_column':
            stdout.write('\tColumn table not found. Creating...')
            call_command('migrate', 'Columns')
            stdout.write('\tMigration complete')
            stdout.write('\tColumn table was successfully created')
            stdout.write('\tRe-running command...')
            call_command('initialize', '-c')
        else:
            raise e

@staticmethod
def initialize_symbols(stdout):
    stdout.write('Filling in the "Symbol" table...')
    try:
        for column, symbols in SYMBOL_LABELS.items():
            for symbol in symbols:

                # Make sure the column exists
                try:
                    c = Column.objects.get(label=column)
                except (OperationalError, Exception) as e:
                    if str(e) == 'no such table: Columns_column':
                        stdout.write(f'\tColumn "{column}" does not exist.
Stopping...')
                        stdout.write(f'\tPlease run "python manage.py initialize -c" and try again.')
                    return
                else:
                    raise e

                s, created = Symbol.objects.get_or_create(
                    label=symbol['label'],
                    position=symbol['position'],
                    column=c
                )
                if created:
                    stdout.write(f'\tCreated Symbol: "{s.label}"')
                else:
                    stdout.write(f'\tSymbol: "{s.label}" already exists')
    # If the database does not exist, an OperationalError is thrown
    # We want to run migrations automatically
    except (OperationalError, Exception) as e:
        if str(e) == 'no such table: Columns_symbol':
            stdout.write('\tSymbol table not found. Creating...')
```

```
        call_command('migrate', 'Columns')
        stdout.write('\tMigration complete')
        stdout.write('\tSymbol table was successfully created')
        stdout.write('\tRe-running command...')
        call_command('initialize', '-s')
    else:
        raise e

@staticmethod
def initialize_data(stdout):
    stdout.write('Initializing Weather Data...')
    call_command('initiate_locations')
    call_command('initiate_wstates')
```

Tabelle 92 Anhang: Skripte - Initialize.py

Datei wetter/management/commands/initiate_locations.py



```
from django.core.management import BaseCommand, call_command
from django.db import OperationalError

from wetter.models import Location

LOCATIONS = [
    {'name': 'Bern',
     'latitude': 46.9480,
     'longitude': 7.4474,
    },
    {'name': 'Zurich',
     'latitude': 47.3769,
     'longitude': 8.5417,
    },
    {'name': 'London',
     'latitude': 51.5074,
     'longitude': 0.1278,
    },
    {'name': 'Paris',
     'latitude': 48.8566,
     'longitude': 2.3522,
    },
    {'name': 'San Francisco',
     'latitude': 37.7749,
     'longitude': -122.4194,
    }
]

class Command(BaseCommand):
    help = 'Create Locations'

    def handle(self, *args, **options): # pragma: no cover
        self.stdout.write('Creating Locations...')
        try:
            # Create locations
            for location in LOCATIONS:
                city, created = Location.objects.get_or_create(**location)
        except OperationalError:
            self.stderr.write('An error occurred while creating locations')

if __name__ == '__main__':
    Command().handle()
```

```

        if created:
            self.stdout.write(f'\tCreated Location {city.name}
({city.latitude}, {city.longitude})')
        else:
            self.stdout.write(f'\tLocation {city.name} already exists')

    # If the database does not exist or the Tables are not created, an OperationalError is thrown
    # We want to run migrations automatically
    except (OperationalError, Exception) as e:
        if str(e) == 'no such table: wetter_location':
            self.stdout.write('\tLocation table not found. Creating...')
            call_command('migrate', 'wetter')
            self.stdout.write('\tMigration complete')
            self.stdout.write('\tLocation table was successfully created')
            self.stdout.write('\tRe-running command...')
            call_command('initiate_locations')
        else:
            raise e

```

Tabelle 93 Anhang: Skripte - initiate_locations.py

Datei wetter/management/commands/initiate_wstates.py



```

from django.core.management import BaseCommand, call_command
from django.db import OperationalError

from wetter.models import WeatherSymbol

OPEN_WEATHER_MAP_SYMBOLS = [
    {'label': 'clear sky',
     'rating': 9,
    },
    {'label': 'few clouds',
     'rating': 8,
    },
    {'label': 'mist',
     'rating': 7,
    },
    {'label': 'scattered clouds',
     'rating': 6,
    },
    {'label': 'broken clouds',
     'rating': 5,
    },
    {'label': 'rain',
     'rating': 4,
    },
    {'label': 'shower rain',
     'rating': 3,
    },
    {'label': 'thunderstorm',
     'rating': 2,
    },
    {'label': 'snow',
     'rating': 1,
    }
]

```

```

        'rating': 1,
    }

class Command(BaseCommand):
    help = 'Create weather symbols'

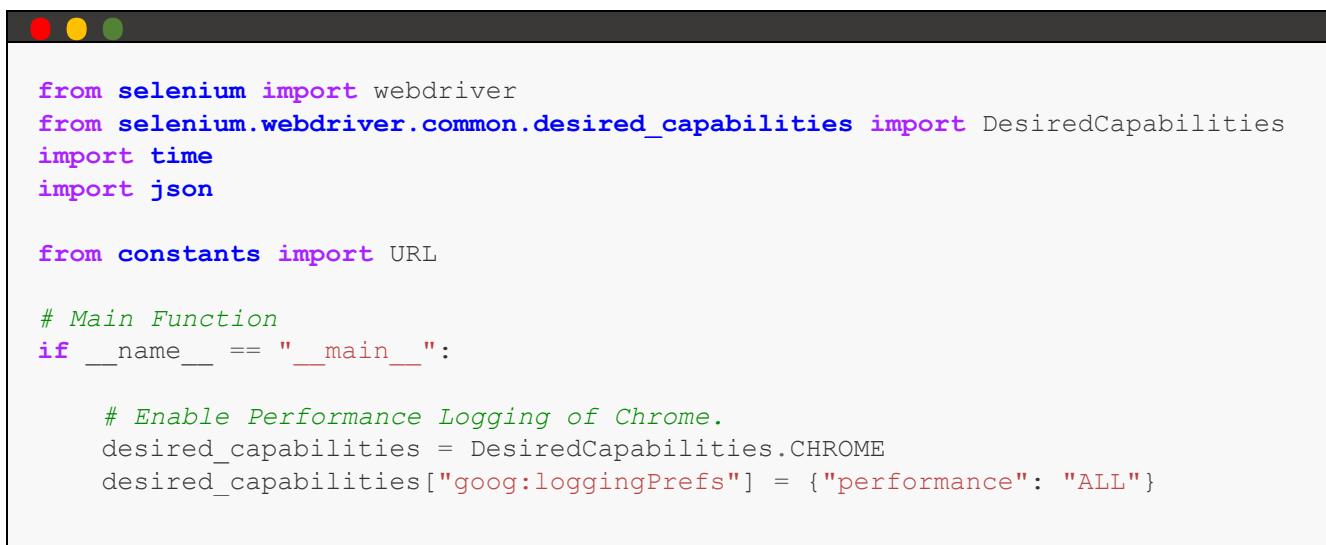
    def handle(self, *args, **options): # pragma: no cover
        self.stdout.write('Creating weather symbols...')
        try:
            # Create symbols
            for symbol in OPEN_WEATHER_MAP_SYMBOLS:
                s, created = WeatherSymbol.objects.get_or_create(**symbol)
                if created:
                    self.stdout.write(f'\tCreated Weather symbol {s.label} '
({s.rating})')
                else:
                    self.stdout.write(f'\tWeather symbol {s.label} ({s.rating}) '
already exists')
            # If the database does not exist or the Tables are not created, an OperationalError is thrown
            # We want to run migrations automatically
            except (OperationalError, Exception) as e:
                if str(e) == 'no such table: wetter_symbol':
                    self.stdout.write('\tweather_symbol table not found. Creating...')
                    call_command('migrate', 'wetter')
                    self.stdout.write('\tMigration complete')
                    self.stdout.write('\tweather_symbol table was successfully created')
                self.stdout.write('\tRe-running command...')
                call_command('initiate_wstates')
            else:
                raise e

```

Tabelle 94 Anhang: Skripte - initiate_wstates.py

20.9.4. Skripte für die Recherche und das Parsing von Daten

Datei wetter/utils/1_get_headers.py



```

from selenium import webdriver
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
import time
import json

from constants import URL

# Main Function
if __name__ == "__main__":
    # Enable Performance Logging of Chrome.
    desired_capabilities = DesiredCapabilities.CHROME
    desired_capabilities["goog:loggingPrefs"] = {"performance": "ALL"}

```

```
# Create the webdriver object and pass the arguments
options = webdriver.ChromeOptions()

# Chrome will start in Headless mode
options.add_argument('headless')

# Ignores any certificate errors if there is any
options.add_argument("--ignore-certificate-errors")

# Chrome driver executable path
options.add_argument("--disable-extensions")

# Startup the chrome webdriver with executable path and
# pass the chrome options and desired capabilities as
# parameters.
driver = webdriver.Chrome(options=options,
                           desired_capabilities=desired_capabilities)

# Send a request to the website and let it load
driver.get(URL)

# Sleeps for 10 seconds to let the page load
time.sleep(3)

# Gets all the logs from performance in Chrome
logs = driver.get_log("performance")

counter = 0
for log in logs:
    network_log = json.loads(log["message"])["message"]

    # Checks if the current 'method' key has any
    # Network request related value
    if "Network.request" in network_log["method"]:
        # Get the params from the current log
        try:
            data_url = network_log["params"]["request"]["url"]
            if 'chartData.precipitation_day.BER.de.json' in data_url:
                headers = network_log["params"]["request"]["headers"]
                print(headers)
        except: # noqa
            # I don't care about the error here
            pass

    print("Quitting Selenium WebDriver")
driver.quit()
```

Tabelle 95 Anhang: Skripte - 1_get_headers.py

Datei wetter/utils/ 2_get_daily_precipitation_data.py



```
import requests
from datetime import datetime

from constants import URL, HEADERS, CITY
```

```
def get_data():
    response = requests.get(URL, headers=HEADERS)
    data = response.json()
    return data

def get_precipitation_data():
    precipitation_data = get_data()["series"][0]["data"]
    return precipitation_data

def get_precipitation_data_for_day(_day):
    precipitation_data = get_precipitation_data()
    return precipitation_data[_day]

if __name__ == '__main__':
    # write data to csv
    length = len(get_precipitation_data())
    with open(f"precipitation_data_{CITY}.csv", "w") as file:
        for day in range(0, length):
            day_data = get_precipitation_data_for_day(day)

            # convert timestamp to date
            timestamp = day_data[0] / 1000
            date = datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d')

            file.write(f"{date}, {day_data[1]}\n")

    print(f"{CITY} precipitation data written to csv")
```

Tabelle 96 Anhang: Skripte - 2_get_daily_precipitation_data.py

Datei wetter/utils/ 3_group_data_by_precipitation.py



```
import json

from constants import CITY

if __name__ == '__main__':

    with open(f"precipitation_data_{CITY}.csv", "r") as file:
        data = file.readlines()
        # group by precipitation amount
        grouped_data = {}
        for line in data:
            line = line.strip().split(",")
            amount = float(amount) / 24
            # round to 3 decimal places
            amount = round(amount, 3)
            if amount not in grouped_data:
```

```

        grouped_data[amount] = []
        grouped_data[amount].append(line[0])

    else:
        grouped_data[amount].append(line[0])

    # write grouped data to json
    with open(f"precipitation_data_{CITY}.json", "w") as f:
        json.dump(grouped_data, f)

print(f"Precipitation data for {CITY} written to json")

```

Tabelle 97 Anhang: Skripte - 3_group_data_by_precipitation.py

Datei wetter/utils/ 4_count_similarities.py



```

import json

from constants import CITY

if __name__ == '__main__':

    with open('precipitation_data_BER.json', 'r') as f:
        data = json.load(f)

    count_similarities = {}
    for key, value in data.items():
        count_similarities[key] = len(value)

    # write similarity of each amount to json
    with open(f"precipitation_data_{CITY}_similarities.json", "w") as f:
        json.dump(count_similarities, f)

    print(f"Finished writing similarities to {CITY}")

```

Tabelle 98 Anhang: Skripte - 4_count_similarities.py

Datei wetter/utils/constants.py



```

BERN = 'BER'  # Bern/Zollikofen
ZURICH = 'REH' # Zurich/Affoltern

CITY = BERN

DOMAIN = 'https://www.meteoschweiz.admin.ch/'
DATA_PATH = f'product/input/measured-values/chartData/precipita-
tion_day/chartData.precipitation_day.{CITY}.de.json'
URL = f'{DOMAIN}{DATA_PATH}'

HEADERS = {
    "referer": f'{DOMAIN}home/messwerte.html?param=messwerte-niederschlag-
1d&station={CITY}&chart=day',
}

```

```
}
```

Tabelle 99 Anhang: Skripte - constants.py

20.9.5. Routing

Nachfolgend sind die Routing-Dateien zu sehen.

Datei core/urls.py

```
"""core URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import: from my_app import views
    2. Add a URL to urlpatterns: path('', views.home, name='home')
Class-based views
    1. Add an import: from other_app.views import Home
    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
"""

from django.contrib import admin
from django.urls import path, include
from django.views.generic import RedirectView

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', RedirectView.as_view(url='/qr/')),
    path('qr/', include('wetter.urls')),
]
```

Tabelle 100 Anhang: Routing - core/urls.py

Datei wetter/urls.py

```
from django.urls import path
from .views import qr_view, is_occupied, show_data

urlpatterns = [
    path('', qr_view, name='qr_view'),
    path('api/is_occupied/', is_occupied, name='is_occupied'),
    path('api/show_data/<int:weather_hourly_id>', show_data, name='show_data'),
]
```

Tabelle 101 Anhang: Routing - wetter/urls.py

20.9.6. Django Admin Einstellung

Im Columns Verzeichnis gibt es weitere Django Admin Einstellungen, aber die wurde von Herrn Baumgartner gemacht.

Folgend sind die Einstellungen (bzw. Quellcode) die ich geschrieben habe.

Datei `wetter/admin.py`

```
from django.contrib import admin
from .models import Location, WeatherSymbol, WeatherHourly, WeatherDaily

@admin.register(Location)
class LocationModelAdmin(admin.ModelAdmin):
    list_display = [
        'name',
        'latitude',
        'longitude'
    ]

    list_filter = [
        'name'
    ]


@admin.register(WeatherSymbol)
class WeatherSymbolModelAdmin(admin.ModelAdmin):
    list_display = [
        'label',
        'rating'
    ]

    list_filter = [
        'rating'
    ]


@admin.register(WeatherHourly)
class WeatherHourlyModelAdmin(admin.ModelAdmin):
    list_display = [
        'location',
        'temp',
        'weather_symbol',
        'rain',
        'trend',
        'date'
    ]

    list_filter = [
        'trend',
        'date'
    ]


@admin.register(WeatherDaily)
class WeatherDailyModelAdmin(admin.ModelAdmin):
    list_display = [
        'date',
        'symbol',
        'location'
    ]
```

Tabelle 102 Anhang: Django Admin - admin.py

20.9.7. Modelle

Datei wetter/models.py

```
from django.db import models

class Location(models.Model):
    """Model for a location."""

    name = models.CharField(
        max_length=100,
        unique=True,
        default="Bern",
        verbose_name="City",
        help_text="Name of the city",
    )
    latitude = models.FloatField(
        default=46.9480,
        verbose_name="Latitude",
        help_text="Latitude of the city",
    )
    longitude = models.FloatField(
        default=7.4474,
        verbose_name="Longitude",
        help_text="Longitude of the city",
    )

    def __str__(self):
        return self.name

class WeatherSymbol(models.Model):
    """Model for a weather symbol."""

    label = models.CharField(
        max_length=50,
        default="",
        verbose_name="Symbol",
        help_text="Weather symbol",
    )
    rating = models.IntegerField(
        default=0,
        verbose_name="Symbol rating",
        help_text="Symbol rating",
    )

    def __str__(self):
        return f"{self.label} ({self.rating})"

class WeatherHourly(models.Model):
```

```
"""Model for a weather hourly forecast."""

date = models.DateTimeField(
    verbose_name="Date", help_text="Date of the forecast"
)
temp = models.FloatField(
    default=0,
    verbose_name="Temperature",
    help_text="Temperature in °C",
)
rain = models.FloatField(
    default=0,
    verbose_name="Rain",
    help_text="Rain in mm",
)
trend = models.SmallIntegerField(
    default=0, # 0 = no change, 1 = rising, 2 = falling
    verbose_name="Trend",
    help_text="Trend of the Weather",
)
weather_symbol = models.ForeignKey(
    WeatherSymbol,
    on_delete=models.CASCADE,
    verbose_name="Symbol",
    help_text="Weather symbol",
)
location = models.ForeignKey(
    Location,
    on_delete=models.CASCADE,
    verbose_name="Location",
    help_text="Location of the forecast",
)

class Meta:
    ordering = ['-date']
    unique_together = ('date', 'location')

def __str__(self):
    return f'{self.date.strftime("%Y-%m-%d")} {self.location.name}'


class WeatherDaily(models.Model):
    """Model for a weather daily forecast."""

date = models.DateTimeField(
    verbose_name="Date", help_text="Date of the forecast"
)
symbol = models.ForeignKey(
    WeatherSymbol,
    on_delete=models.CASCADE,
    verbose_name="Symbol",
    help_text="Weather symbol",
)
location = models.ForeignKey(
    Location,
    on_delete=models.CASCADE,
    verbose_name="Location",
    help_text="Location of the forecast",
```

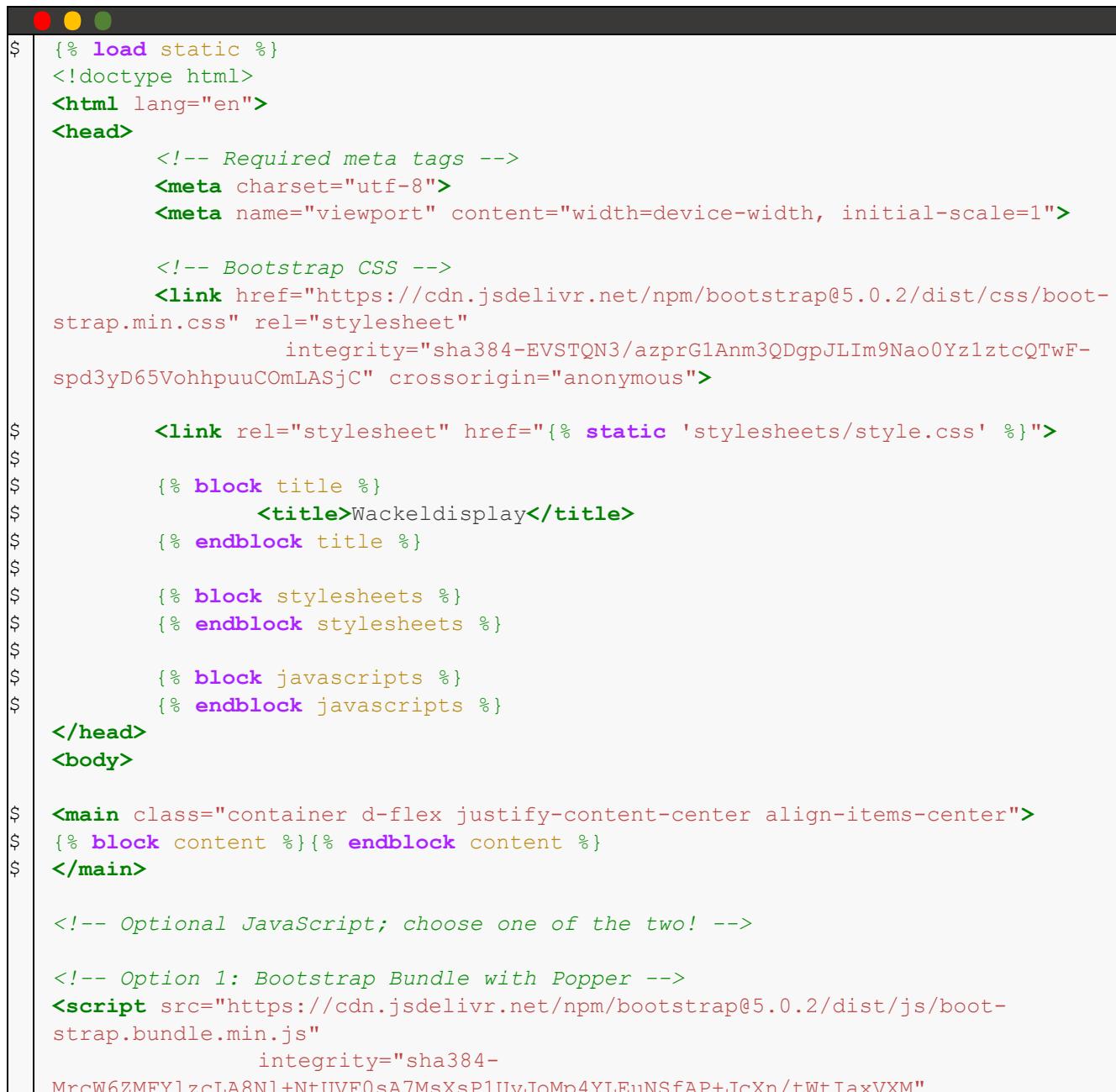
```
)\n\n    class Meta:\n        ordering = ['-date']\n        unique_together = ('date', 'location')\n\n        def __str__(self):\n            return f'{self.date.strftime('%Y-%m-%d')} {self.location.name}\n{self.symbol.rating}'
```

Tabelle 103 Anhang: Models - models.py

20.9.8. Templates (Views)

Nachfolgend sind die Templates-Dateien zu sehen.

Datei templates/base.html



The screenshot shows a code editor window with a dark theme. The title bar has three colored circles (red, yellow, green). The main area contains the content of the base.html template. The code uses Python's Jinja2 templating syntax, including blocks and variables. It includes Bootstrap CSS imports, a title block, and a main container for the content.

```
{% load static %}\n<!doctype html>\n<html lang="en">\n<head>\n    <!-- Required meta tags -->\n    <meta charset="utf-8">\n    <meta name="viewport" content="width=device-width, initial-scale=1">\n\n    <!-- Bootstrap CSS -->\n    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"\n          integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQT吳F-\n          spd3yD65VohhpuuComLASjC" crossorigin="anonymous">\n\n    <link rel="stylesheet" href="{% static 'stylesheets/style.css' %}">\n\n    {% block title %}\n        <title>Wackeldisplay</title>\n    {% endblock title %}\n\n    {% block stylesheets %}\n    {% endblock stylesheets %}\n\n    {% block javascripts %}\n    {% endblock javascripts %}\n</head>\n<body>\n\n<main class="container d-flex justify-content-center align-items-center">\n    {% block content %}{% endblock content %}\n</main>\n\n<!-- Optional JavaScript; choose one of the two! -->\n\n<!-- Option 1: Bootstrap Bundle with Popper -->\n<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"\n       integrity="sha384-\n       MrcW6ZMFYlzcLA8Nl+NdUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIAxVXM"
```

```
crossorigin="anonymous">></script>

<!-- Option 2: Separate Popper and Bootstrap JS -->
<!--
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsoLX15PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-cVKIPhGWiC2A14u+LWgxKTRIcfu0JTxR+EQDz/bgldo-Ey14H0zUF0QKbrJ0EcQF" crossorigin="anonymous"></script>
-->
</body>
</html>
```

Tabelle 104 Anhang: Templates - base.html

Datei templates/wetter/qr.html



```
{% extends 'base.html' %}
{% load static parse_value %}

{% block javascripts %}
<script>
    let currentDataID = 0;
    let showConfirmModal;
    let closeModal;
    let modalConfirmed;
    document.addEventListener("DOMContentLoaded", function () {
        const _confirmModal = new bootstrap.Modal(document.getElementById('confirmModal'))

        showConfirmModal = (id) => {
            currentDataID = id;

            fetch('/qr/api/is_occupied')
                .then(response => response.json())
                .then(data => {
                    if (data.occupied) {
                        document.querySelector('#confirmModalLabel').innerHTML = 'Wackeldisplay in Betrieb';
                        document.querySelector('#modalBody').innerHTML = 'Wackeldisplay ist gerade in Betrieb, versuchen Sie es später noch einmal.';
                        document.querySelector('#confirmButton').style.display = 'none';
                    } else {
                        document.querySelector('#confirmModalLabel').innerHTML = 'Bestätigen';
                        document.querySelector('#modalBody').innerHTML = 'Wollen Sie wirklich die Daten auf dem Wackeldisplay anzeigen lassen?';
                        document.querySelector('#confirmButton').style.display = 'block';
                    }
                })
        }
    });
</script>

```

```
        })
        .then(() => {
        _confirmModal.show();
    })
    .catch(error => console.error(error));
}

modalConfirmed = () => {
    fetch('/qr/api/show_data/' + currentDataID)
        .then(response => response.json())
        .then(data => {
            if (data.started) {
                _confirmModal.hide();
            } else {
                _confirmModal.hide();
                console.error(data.er-
ror);
            }
        })
        .catch(error => console.error(error));
}

closeModal = () => {
    _confirmModal.hide();
}
});

</script>
{%
  endblock javascripts %}

{%
  block title %
    <title>Wackeldisplay | Wetter</title>
{%
  endblock title %}

{%
  block content %
    <section class="w-100">
        {% if weather_hourly %}
            <div class="table-responsive">
                <table class="table table-bordered table-sm">
                    <thead>
                        <tr class="text-center">
                            <th scope="col">Ort</th>
                            <th scope="col">Temperatur <br>in
                            °C</th>
                            <th scope="col">Zustand</th>
                            <th scope="col">Niederschlag
                            <br>in mm</th>
                            <th scope="col">Trend</th>
                            <th scope="col">Aktion</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for w in weather_hourly %}
                            <tr class="text-center">
                                <th class="text-start"
scope="row">{{ w.location }}</th>
                                <td>{{ w.temp }}</td>

```

```

w.weather_symbol | get_url_ow_symbol }}" alt=""></td>
<td>{{ w.rain }}</td>
{%
  if w.trend == 0 %
    <td></td>
    {%
      elif w.trend == 1 %
        <td></td>
        {%
          else %
            <td></td>
            {%
              endif %
            <td>
              <button type="button" class="btn btn-primary" onclick="showConfirmModal({{ w.id }})">
                Zeigen
              </button>
            </td>
          </tr>
        {%
          endfor %
        </tbody>
      </table>
    </div>
{%
  else %
    <div class="alert alert-danger" role="alert">
      <h4 class="alert-heading">Fehler!</h4>
      <p>Es konnte keine Verbindung zum Server hergestellt werden.</p>
      <hr>
      <p class="mb-0">Bitte wenden Sie sich an die Personal das 89grad GmbH.</p>
    </div>
{%
  endif %
</section>

<!-- Modal Confirm -->
<div class="modal fade" id="confirmModal" tabindex="-1" aria-labelledby="confirmModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="confirmModalLabel">Bestätigen</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" onclick="closeModal()"></button>
      </div>
      <div class="modal-body" id="modalBody">
        Wollen Sie wirklich die Daten auf dem Wackeldisplay anzeigen lassen?
      </div>
      <div class="modal-footer">
        <button id="closeButton" type="button" class="btn btn-secondary" data-bs-dismiss="modal" onclick="closeModal()">Schliessen
      </div>
    </div>
  </div>
</div>

```

```
        </button>
        <button id="confirmButton" type="button"
class="btn btn-primary" onclick="modalConfirmed()">
            Bestätigen
        </button>
    </div>
</div>
</div>
{%
  % endblock content %
}
```

Tabelle 105 Anhang: Templates - wetter/qr.html

20.9.9. Views (Controllers)

Datei wetter/views.py



```
import json
import os
from datetime import datetime

from django.shortcuts import render
from django.http import JsonResponse

from core import settings
from wetter.models import WeatherHourly


def qr_view(request):
    """View for the QR code."""
    weather_hourly = WeatherHourly.objects.all()[:5]
    if weather_hourly:
        with open(settings.MQTT_OCCUPIED, 'w') as f:
            json.dump(datetime.now().timestamp(), f)

        with open(settings.MQTT_CURRENT_DATA, 'w') as f:
            json.dump({
                'show_data': [w.id for w in weather_hourly]
            }, f)
    return render(request, 'wetter/qr.html', {'weather_hourly': weather_hourly})


def is_occupied(request):
    # Check if the demo is occupied
    # If the demo_started.json exists, the demo is occupied
    if demo_occupied():
        return JsonResponse({'occupied': True})
    else:
        return JsonResponse({'occupied': False})


def show_data(request, weather_hourly_id):
    # Start demo
    if request.method == 'GET':
```

```
# Check if the demo is occupied
# If the demo_started.json exists, the demo is occupied
if demo_occupied():
    return JsonResponse({'started': False})
else:
    # Write the demo_started.json
    with open(settings.MQTT_OCCUPIED, 'w') as f:
        json.dump(datetime.now().timestamp(), f)
    try:
        weather_hourly = WeatherHourly.objects.get(id=weather_hourly_id)
    except WeatherHourly.DoesNotExist:
        return JsonResponse({'started': False})

    with open(settings.MQTT_CURRENT_DATA, 'w') as f:
        json.dump({
            'show_data': [weather_hourly.id]
        }, f)
    return JsonResponse({'started': True})

def demo_occupied():
    # Check if the demo is occupied
    # If the demo_started.json exists, the demo is occupied
    if os.path.isfile(settings.MQTT_OCCUPIED):
        return True
    else:
        return False
```

Tabelle 106 Anhang: Views - views.py

20.9.10. Hilfeskripte

Hier sind die Skripte die als Hilfsmittel für weiteren Quellcode dienen und nicht zu anderen Kapiteln gehören.

Datei `wetter/constants.py`. Hier werden einige Konstanten definiert

```
from core.settings import OW_API_KEY

OWA_URL = f'https://api.openweathermap.org/data/2.5/onecall?ap-
pid={OW_API_KEY}&'

# OpenWeatherMap API has many groups of weather state ids
# This mapping is used to map the weather state id to a weather symbol
# https://openweathermap.org/weather-conditions

RATING_MAP = [
    ['clear sky', 800],
    ['few clouds', 801],
    ['mist', 701, 711, 721, 731, 741, 751, 761, 762, 771, 781],
    ['scattered clouds', 802],
    ['broken clouds', 803, 804],
    ['rain', 500, 501, 502, 503, 504],
```

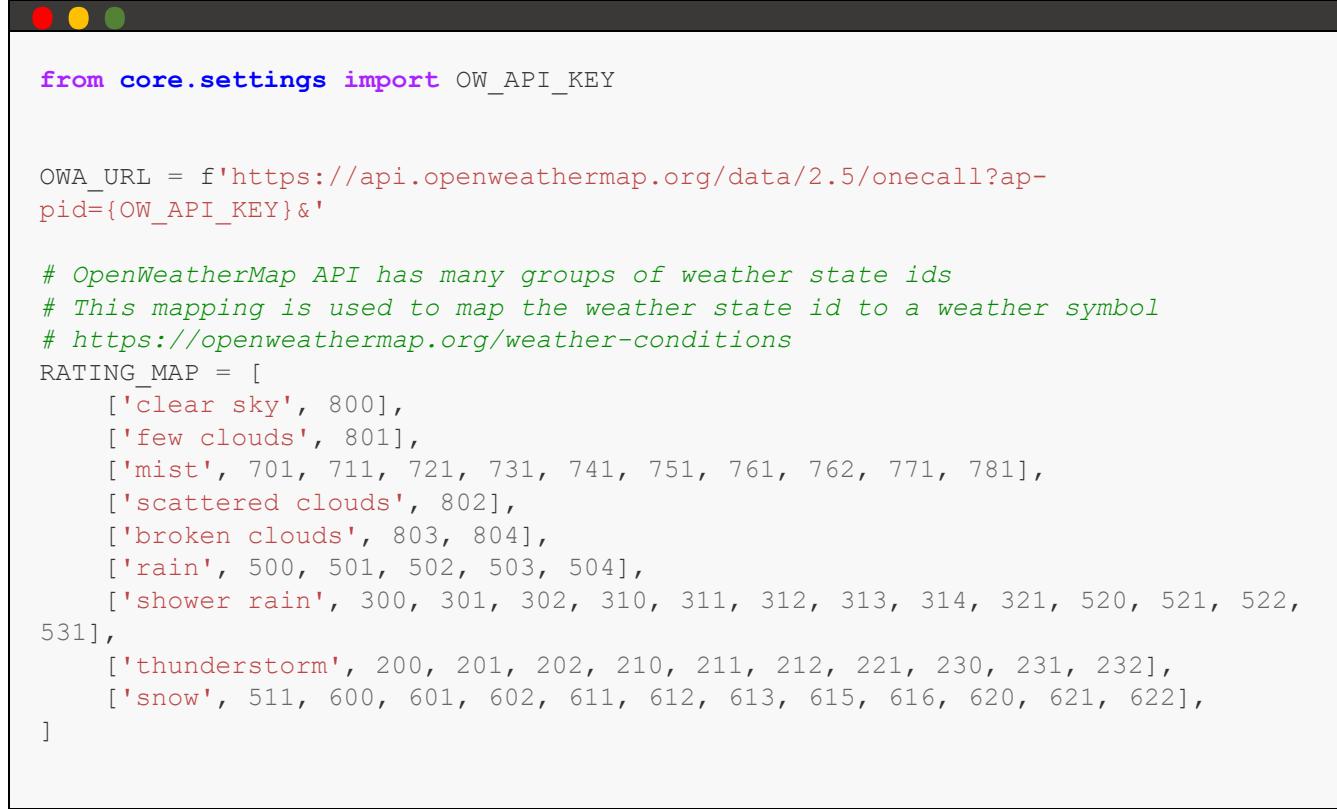
```

['shower rain', 300, 301, 302, 310, 311, 312, 313, 314, 321, 520, 521, 522,
531],
['thunderstorm', 200, 201, 202, 210, 211, 212, 221, 230, 231, 232],
['snow', 511, 600, 601, 602, 611, 612, 613, 615, 616, 620, 621, 622],
]

```

Tabelle 107 Anhang: Hilfeskripte - constants.py

Datei wetter/utils/ow_api_service.py



```

from core.settings import OW_API_KEY

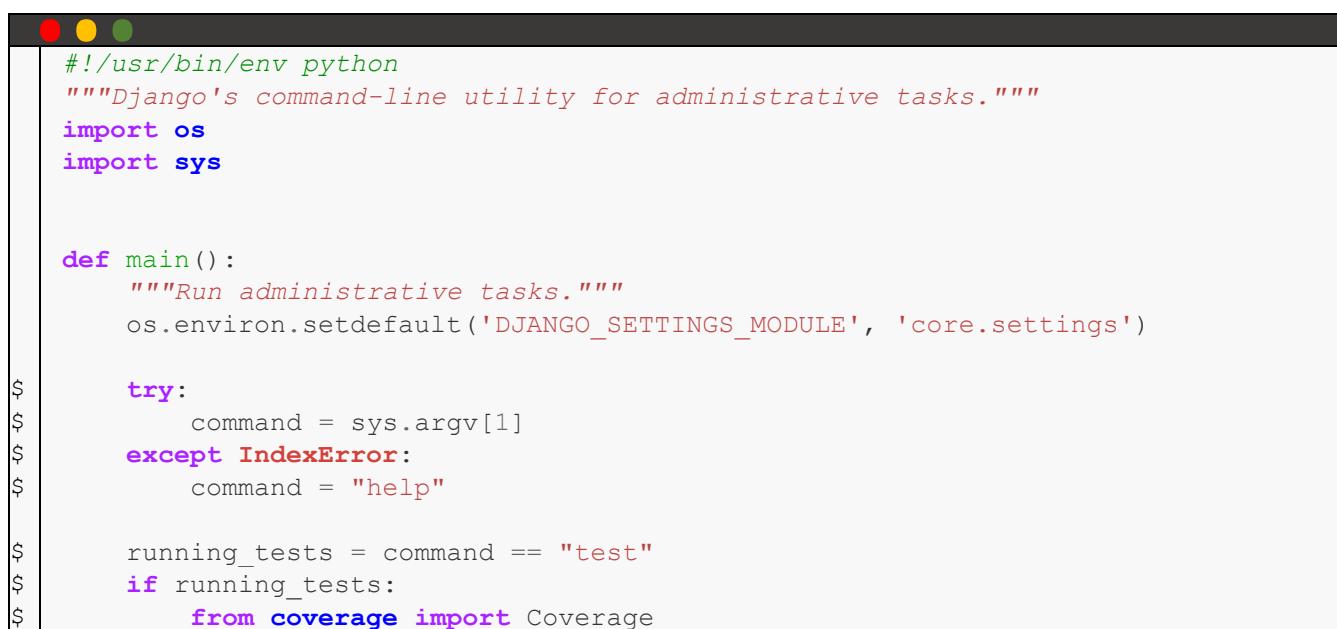
OWA_URL = f'https://api.openweathermap.org/data/2.5/onecall?ap-
pid={OW_API_KEY}&'

# OpenWeatherMap API has many groups of weather state ids
# This mapping is used to map the weather state id to a weather symbol
# https://openweathermap.org/weather-conditions
RATING_MAP = [
    ['clear sky', 800],
    ['few clouds', 801],
    ['mist', 701, 711, 721, 731, 741, 751, 761, 762, 771, 781],
    ['scattered clouds', 802],
    ['broken clouds', 803, 804],
    ['rain', 500, 501, 502, 503, 504],
    ['shower rain', 300, 301, 302, 310, 311, 312, 313, 314, 321, 520, 521, 522,
531],
    ['thunderstorm', 200, 201, 202, 210, 211, 212, 221, 230, 231, 232],
    ['snow', 511, 600, 601, 602, 611, 612, 613, 615, 616, 620, 621, 622],
]

```

Tabelle 108 Anhang: Hilfeskripte - ow_api_service.py

Datei manage.py



```

#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')

    try:
        command = sys.argv[1]
    except IndexError:
        command = "help"

    if running_tests:
        from coverage import Coverage

```

```
$ cov = Coverage()
$c cov.erase()
$c cov.start()
$ try:
$     from django.core.management import execute_from_command_line
$ except ImportError as exc:
$     raise ImportError(
$         "Couldn't import Django. Are you sure it's installed and "
$         "available on your PYTHONPATH environment variable? Did you "
$         "forget to activate a virtual environment?"
$     ) from exc
execute_from_command_line(sys.argv)

$ if running_tests:
$     cov.stop()
$     cov.save()
$     covered = cov.report()
$     if covered < 90:
$         sys.exit(1)

$ if __name__ == '__main__':
main()
```

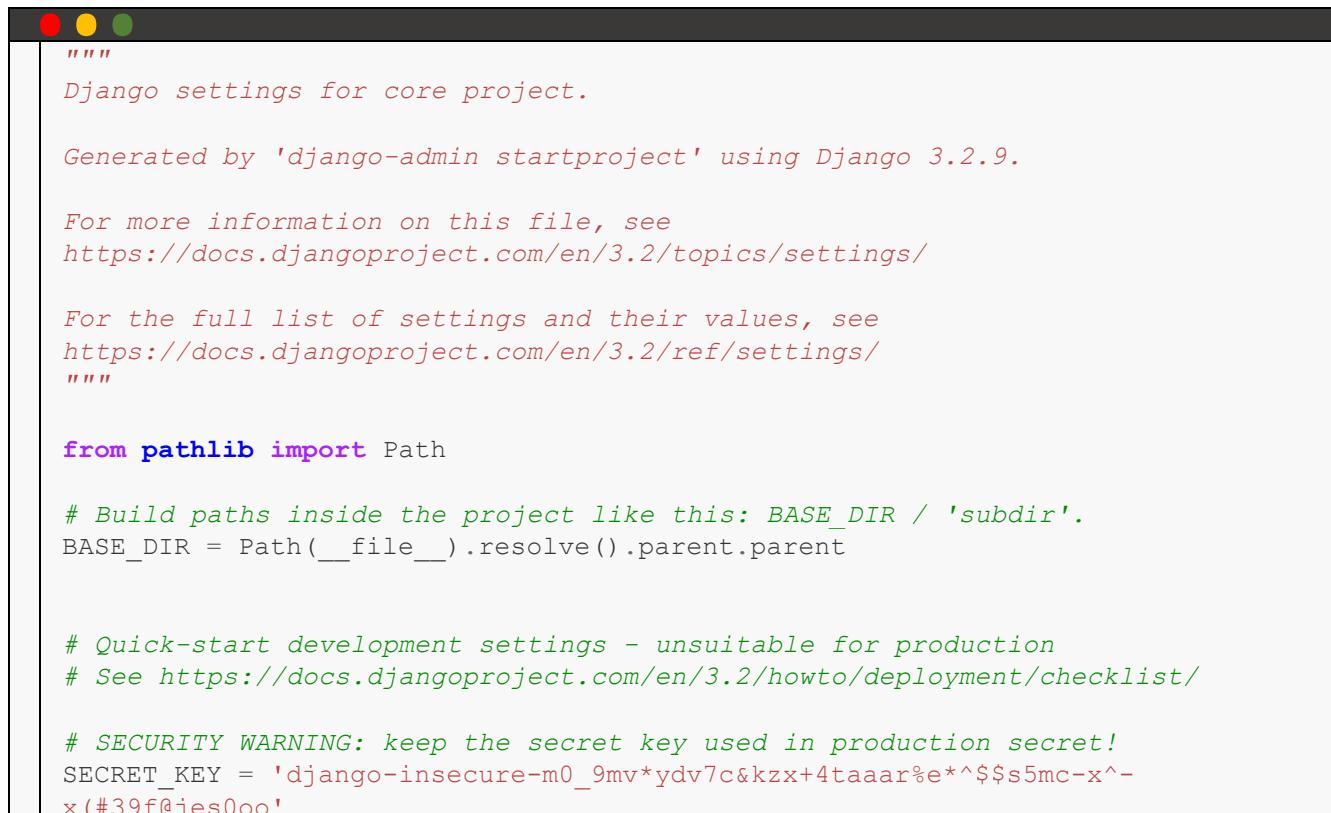
Tabelle 109 Anhang: Hilfeskripte - manage.py

Cronjob Kofigurationsdatei

Datei

20.9.11. Django Settings

Datei core/settings_devel.py



```
"""
Django settings for core project.

Generated by 'django-admin startproject' using Django 3.2.9.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-m0_9mv*ydv7c&kzx+4taaar%e*^$$s5mc-x^-x(#39f@jes0oo'
```

```
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'Columns',
    'wetter',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'core.urls'

TEMPLATES = [
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [
        BASE_DIR / 'templates',
    ],
    'APP_DIRS': True,
    'OPTIONS': {
        'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
        ],
    },
},
]

WSGI_APPLICATION = 'core.wsgi.application'

# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

```
    }

# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

$ STATIC_URL = '/static/'
$ STATICFILES_DIRS = [
$     BASE_DIR / "static",
$ ]
$ STATIC_ROOT = BASE_DIR / "staticfiles"

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

# logging stuff
LOG_DIR = BASE_DIR / 'logs'
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
```

```
'formatters': {
    'simple': {'format': '%(asctime)s %(module)s %(levelname)s %(message)s'},
},
'handlers': {
    'console': {
        'level': 'DEBUG',
        'class': 'logging.StreamHandler',
    },
    'mqtt_file': {
        'level': 'DEBUG',
        'class': 'logging.FileHandler',
        'filename': LOG_DIR / 'mqtt.log',
        'formatter': 'simple',
    },
    'pull_weather': {
        'level': 'DEBUG',
        'class': 'logging.FileHandler',
        'filename': LOG_DIR / 'pull_weather.log',
        'formatter': 'simple',
    },
    'new_symbols': {
        'level': 'DEBUG',
        'class': 'logging.FileHandler',
        'filename': LOG_DIR / 'new_symbols.log',
        'formatter': 'simple',
    },
},
'loggers': {
    'mqtt': {
        'handlers': ['mqtt_file', 'console'],
        'level': 'DEBUG',
    },
    'pull_weather': {
        'handlers': ['pull_weather', 'console'],
        'level': 'DEBUG',
    },
    'new_symbols': {
        'handlers': ['new_symbols', 'console'],
        'level': 'DEBUG',
    },
},
# Fix for missing template variable KeyError: 'is_popup' --> it's not
# an error it's a debug message
# see: https://stackoverflow.com/questions/34797884/getting-error-with-is-popup-variable-in-django-1-9
'django.template': {
    'handlers': ['console'],
    'level': 'INFO',
    'propagate': True,
},
},
}

# mqtt settings
MQTT_TOPICBASE = '/newlobby/columns'
MQTT_STATUSTOPIC = MQTT_TOPICBASE + '/status'
MQTT_USERNAME = 'newlobby_columns_master'
MQTT_PASSWORD = 'd09f5dbdc0625146bd3c2b3ba488a794'
MQTT_CAFILE = BASE_DIR / 'certs/ca.pem'
MQTT_PORT = 8883
```

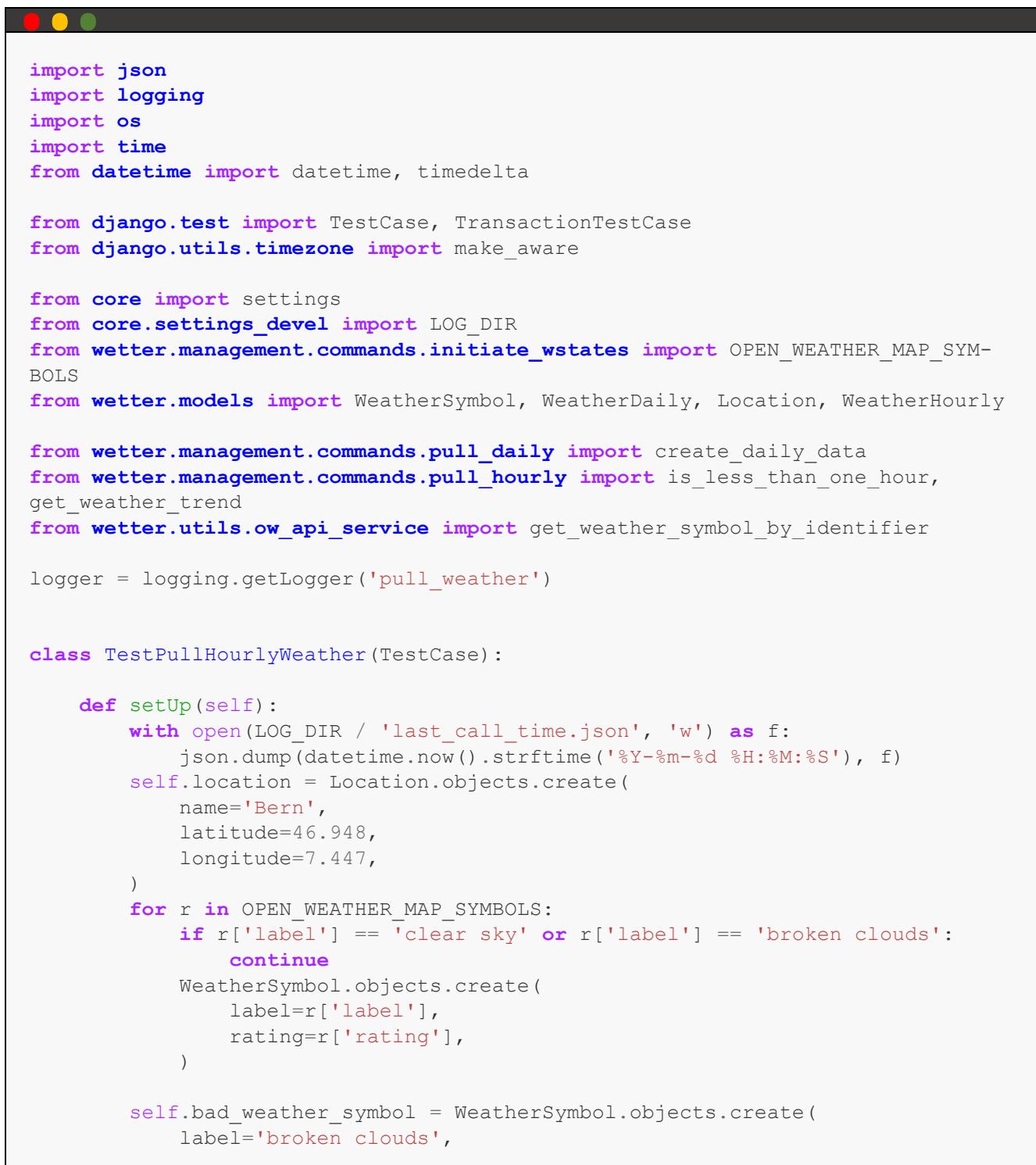
```
$ MQTT_HOST = 'mqtt.e12'
$ MQTT_OCCUPIED = BASE_DIR / 'tmp/demo_started.json'
$ MQTT_CURRENT_DATA = BASE_DIR / 'tmp/current_data.json'
$ OW_API_KEY = 'a761282180b2d0198ccdac27eac5da85'
$ TEST_DATA_DIR = BASE_DIR / 'wetter/test_data'
```

Tabelle 110 Anhang: Django Settings - settings_devel.py

20.9.12. Tests

Nachfolgend ist das Quellcode für Unit-Tests zu finden:

Datei wetter/tests.py



```
import json
import logging
import os
import time
from datetime import datetime, timedelta

from django.test import TestCase, TransactionTestCase
from django.utils.timezone import make_aware

from core import settings
from core.settings_devel import LOG_DIR
from wetter.management.commands.initiate_wstates import OPEN_WEATHER_MAP_SYMBOLS
from wetter.models import WeatherSymbol, WeatherDaily, Location, WeatherHourly

from wetter.management.commands.pull_daily import create_daily_data
from wetter.management.commands.pull_hourly import is_less_than_one_hour,
get_weather_trend
from wetter.utils.ow_api_service import get_weather_symbol_by_identifier

logger = logging.getLogger('pull_weather')

class TestPullHourlyWeather(TestCase):

    def setUp(self):
        with open(LOG_DIR / 'last_call_time.json', 'w') as f:
            json.dump(datetime.now().strftime('%Y-%m-%d %H:%M:%S'), f)
        self.location = Location.objects.create(
            name='Bern',
            latitude=46.948,
            longitude=7.447,
        )
        for r in OPEN_WEATHER_MAP_SYMBOLS:
            if r['label'] == 'clear sky' or r['label'] == 'broken clouds':
                continue
            WeatherSymbol.objects.create(
                label=r['label'],
                rating=r['rating'],
            )

        self.bad_weather_symbol = WeatherSymbol.objects.create(
            label='broken clouds',
```

```
        rating=5
    )
    self.good_weather_symbol = WeatherSymbol.objects.create(
        label='clear sky',
        rating=9
    )

    def test_if_the_pull_was_less_than_an_hour_ago(self):
        self.assertTrue(is_less_than_one_hour())

    def test_get_symbol_by_identifier(self):
        identifier = 800
        from wetter.management.commands.pull_hourly import get_weather_symbol_by_identifier
        self.assertEqual(get_weather_symbol_by_identifier(identifier).label,
                         'clear sky')

    def test_if_the_pull_was_more_than_an_hour_ago(self):
        with open(LOG_DIR / 'last_call_time.json', 'w') as f:
            json.dump((datetime.now() - timedelta(hours=1)).strftime('%Y-%m-%d %H:%M:%S'), f)
        self.assertFalse(is_less_than_one_hour())

    def test_exception_if_less_than_one_hour(self):
        # remove LOG_DIR / 'last_call_time.json' if exists
        if os.path.isfile(LOG_DIR / 'last_call_time.json'):
            os.remove(LOG_DIR / 'last_call_time.json')
        with self.assertLogs(logger='pull_weather') as cm:
            is_less_than_one_hour()
            self.assertIn(cm.output[0], [
                'INFO:pull_weather:Creating empty last_call_time.json'
            ])
            self.assertIn(cm.output[1], [
                'CRITICAL:pull_weather:error parsing json file'
            ])
            self.assertIn(cm.output[2], [
                'CRITICAL:pull_weather:last_call_time.json file is corrupted.'
            ])
        Continuing.
        self.assertFalse(is_less_than_one_hour())

    def test_get_weather_trend_returns_none(self):
        self.assertNone(get_weather_trend(make_aware(datetime.now()), 'clear sky', self.location))

    def test_get_weather_trend_returns_trend_equal(self):
        for i in range(0, 4):
            WeatherDaily.objects.create(
                location=self.location,
                symbol=self.good_weather_symbol,
                date=make_aware(datetime.now() + timedelta(days=i)),
            )
        self.assertEqual(get_weather_trend(make_aware(datetime.now()), self.good_weather_symbol, self.location), 0)

    def test_get_weather_trend_returns_trend_up(self):
        for i in range(1, 4):
```

```
        WeatherDaily.objects.create(
            location=self.location,
            symbol=self.good_weather_symbol,
            date=make_aware(datetime.now() + timedelta(days=i)),
        )
        self.assertEqual(get_weather_trend(make_aware(datetime.now())),
self.bad_weather_symbol, self.location), 1)

    def test_get_weather_trend_returns_trend_down(self):
        for i in range(1, 4):
            WeatherDaily.objects.create(
                location=self.location,
                symbol=self.bad_weather_symbol,
                date=make_aware(datetime.now() + timedelta(days=i)),
            )
            self.assertEqual(get_weather_trend(make_aware(datetime.now())),
self.good_weather_symbol, self.location), 2)

class TestPullDailyWeather(TestCase):

    def setUp(self):
        # Create Location
        self.location = Location.objects.create(
            name='Bern',
            latitude=46.948,
            longitude=7.447,
        )
        # Create WeatherSymbol
        from wetter.models import WeatherSymbol
        self.weather_symbol = WeatherSymbol.objects.create(
            label='clear sky',
            rating=9
        )
        # Load test data
        with open(settings.TEST_DATA_DIR / 'test_data.json', 'r') as file:
            self.test_data = json.load(file)

    def test_first_daily_data_creation(self):
        # Create daily data
        create_daily_data(self.test_data, self.location)
        daily_weather = WeatherDaily.objects.all().count()
        self.assertEqual(daily_weather, 3)

    def test_rest_daily_data_creation(self):
        for i in range(1, 4):
            # Create daily data
            WeatherDaily.objects.create(
                location=self.location,
                date=datetime.now() - timedelta(days=i + 1),
                symbol=self.weather_symbol,
            )

            create_daily_data(self.test_data, self.location)
            daily_weather = WeatherDaily.objects.all().count()
            self.assertEqual(daily_weather, 4)

    def test_create_daily_data_returns_none(self):
```

```
for i in range(1, 4):
    # Create daily data
    WeatherDaily.objects.create(
        location=self.location,
        date=datetime.now() + timedelta(days=i + 1),
        symbol=self.weather_symbol,
    )
self.assertIsNone(create_daily_data(self.test_data, self.location))

class TestPullDailyLogOutput(TransactionTestCase):

    def setUp(self):
        # Create Location
        self.location = Location.objects.create(
            name='Bern',
            latitude=46.948,
            longitude=7.447,
        )
        # Create WeatherSymbol
        from wetter.models import WeatherSymbol
        self.weather_symbol = WeatherSymbol.objects.create(
            label='clear sky',
            rating=9
        )

    def test_exception_if_date_and_location_are_not_unique(self):
        # Load test data
        with open(settings.TEST_DATA_DIR / 'test_corrupted_data.json', 'r') as file:
            self.test_data = json.load(file)

        with self.assertLogs(logger='pull_weather') as cm:
            create_daily_data(self.test_data, self.location)
            self.assertIn(cm.output[0], [
                'INFO:pull_weather:Daily data already exists for Bern on 2022-05-19 11:00:00+00:00'
            ])

class TestModels(TestCase):

    def setUp(self):
        self.date_time = make_aware(datetime.now())
        # Create Location
        self.location = Location.objects.create(
            name='Bern',
            latitude=46.948,
            longitude=7.447,
        )
        # Create WeatherSymbol
        self.weather_symbol = WeatherSymbol.objects.create(
            label='clear sky',
            rating=9
        )
        # Create WeatherDaily
        self.weather_daily = WeatherDaily.objects.create(
            location=self.location,
```

```
        date=self.date_time,
        symbol=self.weather_symbol,
    )
    # Create WeatherHourly
    self.weather_hourly = WeatherHourly.objects.create(
        location=self.location,
        date=self.date_time,
        weather_symbol=self.weather_symbol,
    )

    def test_weather_daily_str(self):
        # WeatherDaily: 2022-05-19 12:11:07.349040+00:00 Bern 9
        self.assertEqual(str(self.weather_daily),
f'{self.date_time.strftime("%Y-%m-%d") } Bern 9')

    def test_weather_hourly_str(self):
        self.assertEqual(str(self.weather_hourly),
f'{self.date_time.strftime("%Y-%m-%d") } Bern')

    def test_location_str(self):
        self.assertEqual(str(self.location), 'Bern')

    def test_weather_symbol_str(self):
        self.assertEqual(str(self.weather_symbol), 'clear sky (9)')

class TestScripts(TestCase):

    def test_open_weather_introduced_new_symbol(self):
        _logger = logging.getLogger('new_symbols')
        with self.assertLogs(logger=_logger) as cm:
            get_weather_symbol_by_identifier(805)
            self.assertIn(cm.output[0], [
                'ERROR:new_symbols:Could not find weather symbol 805'
            ])
            self.assertIn(cm.output[1], [
                'INFO:new_symbols:Please make sure that all weather symbols are
in the database'
            ])
            self.assertIn(cm.output[2], [
                'INFO:new_symbols:In case that Open Weather Map introduces a
new symbol, '
                'please add it to the RATING_MAP in wetter/constants.py'
            ])
            self.assertIn(cm.output[3], [
                'INFO:new_symbols:You can find the list of all weather symbols
',
                'here: https://openweathermap.org/weather-conditions'
            ])
            self.assertIn(cm.output[4], [
                'WARNING:new_symbols:New Symbol ID: 805'
            ])

    class TestViews(TestCase):
        def setUp(self):
            self.location = Location.objects.create(
                name='Bern',
```

```
        latitude=46.948,
        longitude=7.447,
    )
    self.weather_symbol = WeatherSymbol.objects.create(
        label='clear sky',
        rating=9
    )
    for i in range(1, 10):
        # Create WeatherHourly
        WeatherHourly.objects.create(
            location=self.location,
            date=make_aware(datetime.now() - timedelta(hours=i)),
            weather_symbol=self.weather_symbol,
            rain=0,
            trend=0,
        )

def test_qr_view(self):
    response = self.client.get('/qr/')
    self.assertEqual(response.status_code, 200)

def test_qr_view_creates_json(self):
    response = self.client.get('/qr/')
    self.assertEqual(response.status_code, 200)
    # Assert true that file exists
    self.assertTrue(os.path.exists(settings.MQTT_OCCUPIED))

def test_qr_view_creates_json_with_current_data(self):
    response = self.client.get('/qr/')
    self.assertEqual(response.status_code, 200)
    # Assert true that file exists
    self.assertTrue(os.path.exists(settings.MQTT_CURRENT_DATA))

def test_qr_view_return_template(self):
    response = self.client.get('/qr/')
    self.assertTemplateUsed(response, 'wetter/qr.html')

def test_if_is_occupied_returns_json_with_true(self):
    with open(settings.MQTT_OCCUPIED, 'w') as f:
        json.dump(datetime.now().timestamp(), f)
    response = self.client.get('/qr/api/is_occupied/')
    self.assertEqual(response.status_code, 200)
    self.assertEqual(response.json(), {'occupied': True})

def test_if_is_occupied_returns_json_with_false(self):
    # Remove settings.MQTT_OCCUPIED
    time.sleep(1)
    try:
        os.remove(settings.MQTT_OCCUPIED)
    except FileNotFoundError:
        pass
    response = self.client.get('/qr/api/is_occupied/')
    self.assertEqual(response.status_code, 200)
    self.assertEqual(response.json(), {'occupied': False})
```

Tabelle 111 Anhang: Tests - tests.py

20.9.13. Testdaten

Datei wetter/test_data/test_data.json



```
{  
    "lat": 46.948,  
    "lon": 7.4474,  
    "timezone": "Europe/Zurich",  
    "timezone_offset": 7200,  
    "current": {  
        "dt": 1652951690,  
        "sunrise": 1652932245,  
        "sunset": 1652986971,  
        "temp": 24.09,  
        "feels_like": 23.83,  
        "pressure": 1024,  
        "humidity": 49,  
        "dew_point": 12.72,  
        "uvi": 4.65,  
        "clouds": 0,  
        "visibility": 10000,  
        "wind_speed": 2.68,  
        "wind_deg": 279,  
        "wind_gust": 4.47,  
        "weather": [  
            {  
                "id": 800,  
                "main": "Clear",  
                "description": "clear sky",  
                "icon": "01d"  
            }  
        ]  
    },  
    "daily": [  
        {  
            "dt": 1652958000,  
            "sunrise": 1652932245,  
            "sunset": 1652986971,  
            "moonrise": 1652912700,  
            "moonset": 1652941200,  
            "moon_phase": 0.63,  
            "temp": {  
                "day": 24.91,  
                "min": 14.24,  
                "max": 27.11,  
                "night": 14.47,  
                "eve": 17.25,  
                "morn": 17.05  
            },  
            "feels_like": {  
                "day": 24.74,  
                "night": 14.43,  
                "eve": 17.41,  
                "morn": 16.98  
            },  
            "pressure": 1023,  
            "humidity": 49,  
            "clouds": 0,  
            "wind": {  
                "speed": 2.68,  
                "deg": 279,  
                "gust": 4.47  
            },  
            "visibility": 10000,  
            "uv": 4.65  
        }  
    ]  
}
```

```
"dew_point": 13.47,
"wind_speed": 4.03,
"wind_deg": 264,
"wind_gust": 6.98,
"weather": [
    {
        "id": 501,
        "main": "Rain",
        "description": "moderate rain",
        "icon": "10d"
    }
],
"clouds": 40,
"pop": 0.96,
"rain": 6.72,
"uvi": 7.06
},
{
    "dt": 1653044400,
    "sunrise": 1653018581,
    "sunset": 1653073443,
    "moonrise": 1653002700,
    "moonset": 1653032040,
    "moon_phase": 0.66,
    "temp": {
        "day": 27.05,
        "min": 13.37,
        "max": 28.48,
        "night": 18.33,
        "eve": 25.55,
        "morn": 14.42
    },
    "feels_like": {
        "day": 27.28,
        "night": 18.18,
        "eve": 25.62,
        "morn": 14.32
    },
    "pressure": 1021,
    "humidity": 47,
    "dew_point": 13.99,
    "wind_speed": 4.88,
    "wind_deg": 239,
    "wind_gust": 13.06,
    "weather": [
        {
            "id": 500,
            "main": "Rain",
            "description": "light rain",
            "icon": "10d"
        }
    ],
    "clouds": 26,
    "pop": 0.91,
    "rain": 1.6,
    "uvi": 7.43
},
{
```

```
"dt": 1653130800,  
"sunrise": 1653104919,  
"sunset": 1653159913,  
"moonrise": 1653091800,  
"moonset": 1653123240,  
"moon_phase": 0.7,  
"temp": {  
    "day": 23.95,  
    "min": 13.42,  
    "max": 26.31,  
    "night": 15.48,  
    "eve": 22.66,  
    "morn": 14.56  
},  
"feels_like": {  
    "day": 23.89,  
    "night": 15.51,  
    "eve": 22.94,  
    "morn": 14.5  
},  
"pressure": 1021,  
"humidity": 57,  
"dew_point": 14.2,  
"wind_speed": 2.32,  
"wind_deg": 248,  
"wind_gust": 6.83,  
"weather": [  
    {  
        "id": 500,  
        "main": "Rain",  
        "description": "light rain",  
        "icon": "10d"  
    }  
],  
"clouds": 79,  
"pop": 0.85,  
"rain": 1.49,  
"uvi": 6.39  
},  
{  
    "dt": 1653217200,  
    "sunrise": 1653191259,  
    "sunset": 1653246383,  
    "moonrise": 1653180180,  
    "moonset": 1653214380,  
    "moon_phase": 0.75,  
    "temp": {  
        "day": 27.83,  
        "min": 13.06,  
        "max": 27.83,  
        "night": 16.49,  
        "eve": 21.14,  
        "morn": 16.32  
    },  
    "feels_like": {  
        "day": 28.02,  
        "night": 16.73,  
        "eve": 21.48,
```

```
        "morn": 16.28
    },
    "pressure": 1012,
    "humidity": 47,
    "dew_point": 14.83,
    "wind_speed": 2.73,
    "wind_deg": 24,
    "wind_gust": 5.65,
    "weather": [
        {
            "id": 501,
            "main": "Rain",
            "description": "moderate rain",
            "icon": "10d"
        }
    ],
    "clouds": 52,
    "pop": 1,
    "rain": 11,
    "uvi": 7.33
},
{
    "dt": 1653303600,
    "sunrise": 1653277601,
    "sunset": 1653332851,
    "moonrise": 1653268080,
    "moonset": 1653305460,
    "moon_phase": 0.77,
    "temp": {
        "day": 22.89,
        "min": 14.33,
        "max": 22.89,
        "night": 14.76,
        "eve": 16.93,
        "morn": 17.24
    },
    "feels_like": {
        "day": 22.93,
        "night": 14.75,
        "eve": 17.08,
        "morn": 17.4
    },
    "pressure": 1006,
    "humidity": 65,
    "dew_point": 15.16,
    "wind_speed": 5.31,
    "wind_deg": 262,
    "wind_gust": 9.64,
    "weather": [
        {
            "id": 500,
            "main": "Rain",
            "description": "light rain",
            "icon": "10d"
        }
    ],
    "clouds": 95,
    "pop": 1,
```

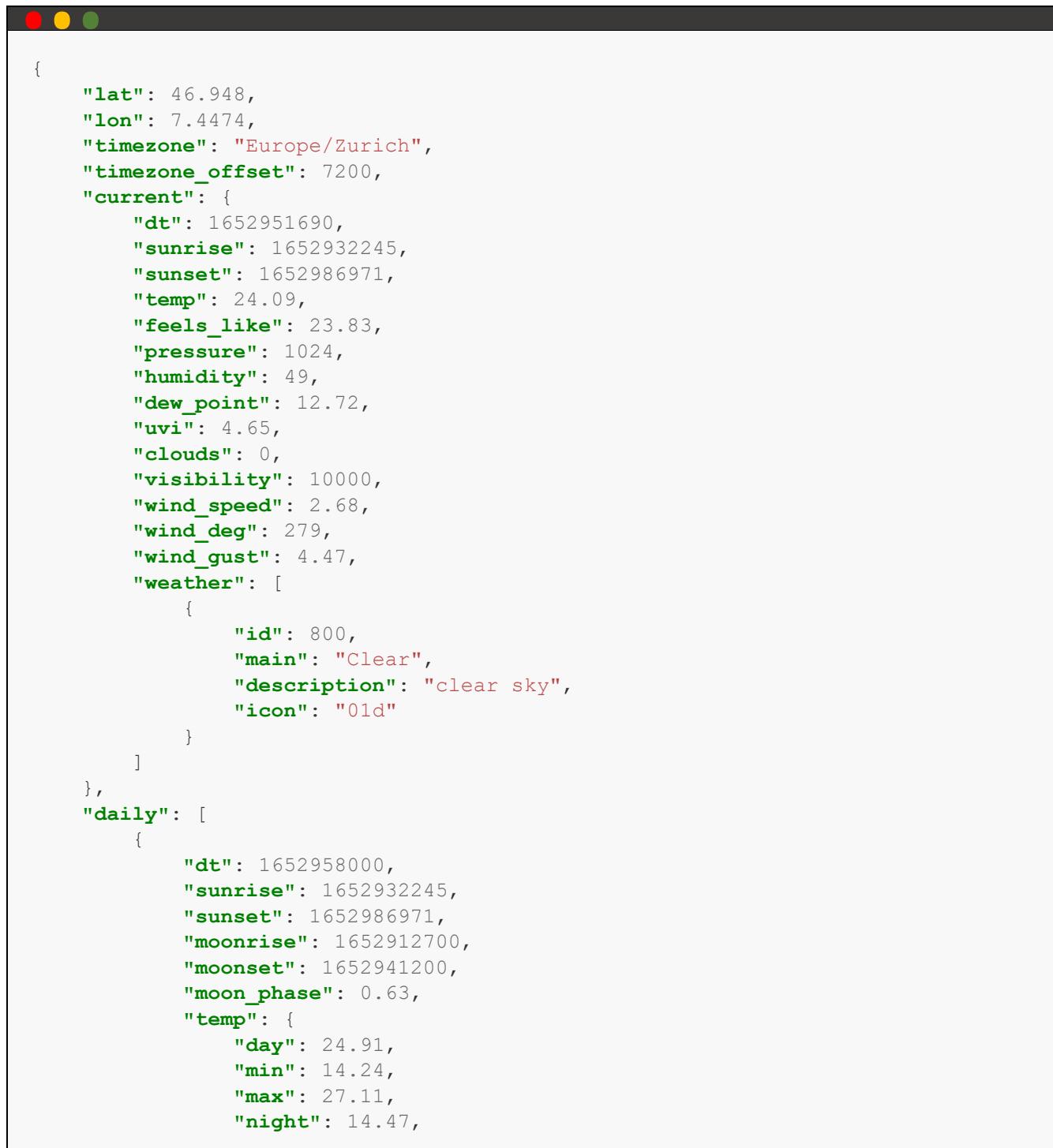
```
    "rain": 4.87,
    "uvi": 7.48
  },
  {
    "dt": 1653390000,
    "sunrise": 1653363945,
    "sunset": 1653419319,
    "moonrise": 1653355740,
    "moonset": 1653396240,
    "moon_phase": 0.81,
    "temp": {
      "day": 16.29,
      "min": 9.98,
      "max": 18.88,
      "night": 9.98,
      "eve": 16.01,
      "morn": 11.59
    },
    "feels_like": {
      "day": 15.57,
      "night": 9.98,
      "eve": 15.52,
      "morn": 11.23
    },
    "pressure": 1014,
    "humidity": 61,
    "dew_point": 7.62,
    "wind_speed": 1.89,
    "wind_deg": 258,
    "wind_gust": 5.65,
    "weather": [
      {
        "id": 500,
        "main": "Rain",
        "description": "light rain",
        "icon": "10d"
      }
    ],
    "clouds": 85,
    "pop": 0.97,
    "rain": 2.13,
    "uvi": 8
  },
  {
    "dt": 1653476400,
    "sunrise": 1653450291,
    "sunset": 1653505785,
    "moonrise": 1653443220,
    "moonset": 1653486900,
    "moon_phase": 0.84,
    "temp": {
      "day": 21.28,
      "min": 8.58,
      "max": 21.28,
      "night": 14.47,
      "eve": 16.42,
      "morn": 12.72
    },
  }
```

```
"feels_like": {
    "day": 20.61,
    "night": 13.96,
    "eve": 16.08,
    "morn": 12.06
},
"pressure": 1015,
"humidity": 44,
"dew_point": 8.03,
"wind_speed": 2.17,
"wind_deg": 274,
"wind_gust": 3.17,
"weather": [
    {
        "id": 804,
        "main": "Clouds",
        "description": "overcast clouds",
        "icon": "04d"
    }
],
"clouds": 96,
"pop": 0.09,
"uvi": 8
},
{
    "dt": 1653562800,
    "sunrise": 1653536639,
    "sunset": 1653592249,
    "moonrise": 1653530640,
    "moonset": 1653577500,
    "moon_phase": 0.87,
    "temp": {
        "day": 23.47,
        "min": 10.11,
        "max": 23.47,
        "night": 15.03,
        "eve": 18.97,
        "morn": 14.29
    },
    "feels_like": {
        "day": 23.13,
        "night": 14.78,
        "eve": 18.93,
        "morn": 13.73
    },
    "pressure": 1020,
    "humidity": 48,
    "dew_point": 11.13,
    "wind_speed": 1.96,
    "wind_deg": 352,
    "wind_gust": 3.37,
    "weather": [
        {
            "id": 500,
            "main": "Rain",
            "description": "light rain",
            "icon": "10d"
        }
    ]
}
```

```
        ],
        "clouds": 12,
        "pop": 0.22,
        "rain": 0.15,
        "uvi": 8
    }
]
}
```

Tabelle 112 Anhang: Testdata - test_data.json

Datei wetter/test_data/corrupted_data.json



A screenshot of a terminal window displaying a JSON object. The JSON structure includes 'current' and 'daily' sections. The 'current' section contains various weather parameters like latitude, longitude, and current temperature. The 'daily' section contains a single daily forecast entry with details such as sunrise, sunset, moonrise, moonset, and temperature ranges for day, min, max, and night.

```
{
  "lat": 46.948,
  "lon": 7.4474,
  "timezone": "Europe/Zurich",
  "timezone_offset": 7200,
  "current": {
    "dt": 1652951690,
    "sunrise": 1652932245,
    "sunset": 1652986971,
    "temp": 24.09,
    "feels_like": 23.83,
    "pressure": 1024,
    "humidity": 49,
    "dew_point": 12.72,
    "uvi": 4.65,
    "clouds": 0,
    "visibility": 10000,
    "wind_speed": 2.68,
    "wind_deg": 279,
    "wind_gust": 4.47,
    "weather": [
      {
        "id": 800,
        "main": "Clear",
        "description": "clear sky",
        "icon": "01d"
      }
    ]
  },
  "daily": [
    {
      "dt": 1652958000,
      "sunrise": 1652932245,
      "sunset": 1652986971,
      "moonrise": 1652912700,
      "moonset": 1652941200,
      "moon_phase": 0.63,
      "temp": {
        "day": 24.91,
        "min": 14.24,
        "max": 27.11,
        "night": 14.47,
        "eve": 24.91
      }
    }
  ]
}
```

```
        "eve": 17.25,
        "morn": 17.05
    },
    "feels_like": {
        "day": 24.74,
        "night": 14.43,
        "eve": 17.41,
        "morn": 16.98
    },
    "pressure": 1023,
    "humidity": 49,
    "dew_point": 13.47,
    "wind_speed": 4.03,
    "wind_deg": 264,
    "wind_gust": 6.98,
    "weather": [
        {
            "id": 501,
            "main": "Rain",
            "description": "moderate rain",
            "icon": "10d"
        }
    ],
    "clouds": 40,
    "pop": 0.96,
    "rain": 6.72,
    "uvi": 7.06
},
{
    "dt": 1652958000,
    "sunrise": 1652932245,
    "sunset": 1652986971,
    "moonrise": 1652912700,
    "moonset": 1652941200,
    "moon_phase": 0.63,
    "temp": {
        "day": 24.91,
        "min": 14.24,
        "max": 27.11,
        "night": 14.47,
        "eve": 17.25,
        "morn": 17.05
    },
    "feels_like": {
        "day": 24.74,
        "night": 14.43,
        "eve": 17.41,
        "morn": 16.98
    },
    "pressure": 1023,
    "humidity": 49,
    "dew_point": 13.47,
    "wind_speed": 4.03,
    "wind_deg": 264,
    "wind_gust": 6.98,
    "weather": [
        {
            "id": 501,
```

```
        "main": "Rain",
        "description": "moderate rain",
        "icon": "10d"
    }
],
"clouds": 40,
"pop": 0.96,
"rain": 6.72,
"uvi": 7.06
},
{
"dt": 1652958000,
"sunrise": 1652932245,
"sunset": 1652986971,
"moonrise": 1652912700,
"moonset": 1652941200,
"moon_phase": 0.63,
"temp": {
    "day": 24.91,
    "min": 14.24,
    "max": 27.11,
    "night": 14.47,
    "eve": 17.25,
    "morn": 17.05
},
"feels_like": {
    "day": 24.74,
    "night": 14.43,
    "eve": 17.41,
    "morn": 16.98
},
"pressure": 1023,
"humidity": 49,
"dew_point": 13.47,
"wind_speed": 4.03,
"wind_deg": 264,
"wind_gust": 6.98,
"weather": [
    {
        "id": 501,
        "main": "Rain",
        "description": "moderate rain",
        "icon": "10d"
    }
],
"clouds": 40,
"pop": 0.96,
"rain": 6.72,
"uvi": 7.06
},
{
"dt": 1652958000,
"sunrise": 1652932245,
"sunset": 1652986971,
"moonrise": 1652912700,
"moonset": 1652941200,
"moon_phase": 0.63,
"temp": {
```

```
        "day": 24.91,
        "min": 14.24,
        "max": 27.11,
        "night": 14.47,
        "eve": 17.25,
        "morn": 17.05
    },
    "feels_like": {
        "day": 24.74,
        "night": 14.43,
        "eve": 17.41,
        "morn": 16.98
    },
    "pressure": 1023,
    "humidity": 49,
    "dew_point": 13.47,
    "wind_speed": 4.03,
    "wind_deg": 264,
    "wind_gust": 6.98,
    "weather": [
        {
            "id": 501,
            "main": "Rain",
            "description": "moderate rain",
            "icon": "10d"
        }
    ],
    "clouds": 40,
    "pop": 0.96,
    "rain": 6.72,
    "uvi": 7.06
},
{
    "dt": 1652958000,
    "sunrise": 1652932245,
    "sunset": 1652986971,
    "moonrise": 1652912700,
    "moonset": 1652941200,
    "moon_phase": 0.63,
    "temp": {
        "day": 24.91,
        "min": 14.24,
        "max": 27.11,
        "night": 14.47,
        "eve": 17.25,
        "morn": 17.05
    },
    "feels_like": {
        "day": 24.74,
        "night": 14.43,
        "eve": 17.41,
        "morn": 16.98
    },
    "pressure": 1023,
    "humidity": 49,
    "dew_point": 13.47,
    "wind_speed": 4.03,
    "wind_deg": 264,
```

```
"wind_gust": 6.98,
"weather": [
    {
        "id": 501,
        "main": "Rain",
        "description": "moderate rain",
        "icon": "10d"
    }
],
"clouds": 40,
"pop": 0.96,
"rain": 6.72,
"uvi": 7.06
},
{
    "dt": 1652958000,
    "sunrise": 1652932245,
    "sunset": 1652986971,
    "moonrise": 1652912700,
    "moonset": 1652941200,
    "moon_phase": 0.63,
    "temp": {
        "day": 24.91,
        "min": 14.24,
        "max": 27.11,
        "night": 14.47,
        "eve": 17.25,
        "morn": 17.05
    },
    "feels_like": {
        "day": 24.74,
        "night": 14.43,
        "eve": 17.41,
        "morn": 16.98
    },
    "pressure": 1023,
    "humidity": 49,
    "dew_point": 13.47,
    "wind_speed": 4.03,
    "wind_deg": 264,
    "wind_gust": 6.98,
    "weather": [
        {
            "id": 501,
            "main": "Rain",
            "description": "moderate rain",
            "icon": "10d"
        }
    ],
    "clouds": 40,
    "pop": 0.96,
    "rain": 6.72,
    "uvi": 7.06
},
{
    "dt": 1652958000,
    "sunrise": 1652932245,
    "sunset": 1652986971,
```

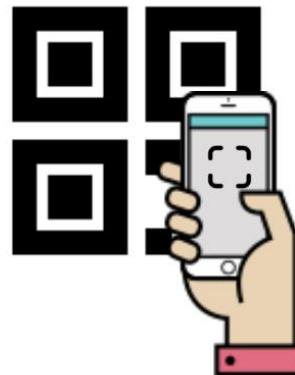
```
"moonrise": 1652912700,  
"moonset": 1652941200,  
"moon_phase": 0.63,  
"temp": {  
    "day": 24.91,  
    "min": 14.24,  
    "max": 27.11,  
    "night": 14.47,  
    "eve": 17.25,  
    "morn": 17.05  
},  
"feels_like": {  
    "day": 24.74,  
    "night": 14.43,  
    "eve": 17.41,  
    "morn": 16.98  
},  
"pressure": 1023,  
"humidity": 49,  
"dew_point": 13.47,  
"wind_speed": 4.03,  
"wind_deg": 264,  
"wind_gust": 6.98,  
"weather": [  
    {  
        "id": 501,  
        "main": "Rain",  
        "description": "moderate rain",  
        "icon": "10d"  
    }  
],  
"clouds": 40,  
"pop": 0.96,  
"rain": 6.72,  
"uvi": 7.06  
},  
{  
    "dt": 1652958000,  
    "sunrise": 1652932245,  
    "sunset": 1652986971,  
    "moonrise": 1652912700,  
    "moonset": 1652941200,  
    "moon_phase": 0.63,  
    "temp": {  
        "day": 24.91,  
        "min": 14.24,  
        "max": 27.11,  
        "night": 14.47,  
        "eve": 17.25,  
        "morn": 17.05  
    },  
    "feels_like": {  
        "day": 24.74,  
        "night": 14.43,  
        "eve": 17.41,  
        "morn": 16.98  
    },  
    "pressure": 1023,
```

```
"humidity": 49,  
"dew_point": 13.47,  
"wind_speed": 4.03,  
"wind_deg": 264,  
"wind_gust": 6.98,  
"weather": [  
    {  
        "id": 501,  
        "main": "Rain",  
        "description": "moderate rain",  
        "icon": "10d"  
    }  
],  
"clouds": 40,  
"pop": 0.96,  
"rain": 6.72,  
"uvi": 7.06  
}  
]  
}
```

Tabelle 113 Anhang: Testdata - corrupted_data.json

Benutzerhandbuch

- Um das Wackeldisplay zu starten, musst du den QR-Code mit dem Handy scannen. (Der QR-Code befindet sich auf dem Wackeldisplay, er ist darauf geklebt).



- Hopla, warte ein bisschen, das Wackeldisplay beginnt gleich. Bis dahin kannst du dir die Daten hier anschauen.

2. Klicke auf die URL

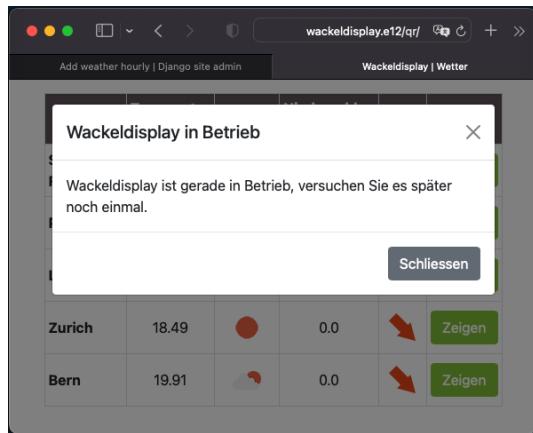
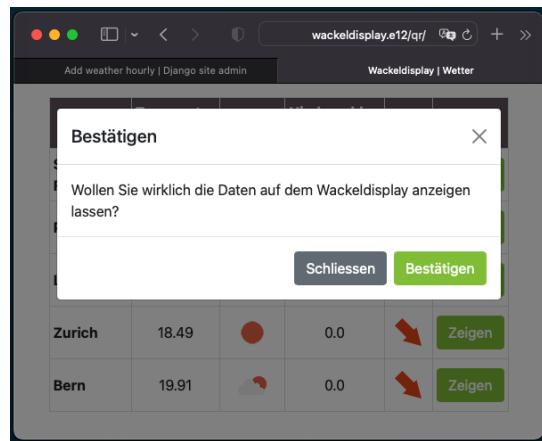
A screenshot of a Django admin interface titled "Wackeldisplay | Wetter". It displays a table with weather data for five cities: San Francisco, Paris, London, Zurich, and Bern. The table includes columns for Ort, Temperatur in °C, Zustand, Niederschlag in mm, Trend, and Aktion. Each row has a green "Zeigen" button with a red arrow icon. Red arrows also point to the "Zeigen" buttons in the screenshot.

Ort	Temperatur in °C	Zustand	Niederschlag in mm	Trend	Aktion
San Francisco	12.31	●	0.0	↗	Zeigen
Paris	16.78	☁️	0.0	↗	Zeigen
London	14.3	☁️	0.0	↗	Zeigen
Zurich	18.49	●	0.0	↗	Zeigen
Bern	19.91	☁️	0.0	↗	Zeigen



- Sobald die Wackeldisplay aufhört (die Demo dauert etwa 3-4 Minuten), kannst du auf die Taste «Zeigen» in der gewünschten Zeile klicken.

5. Aber du musst die Wahl bestätigen.



6. Du kannst eine neue Ausgabe erst starten, wenn die vorherige beendet ist.

FAQ

Wackeldisplay lässt sich nicht starten?

Bist du sicher, dass das Gerät an das Stromnetz angeschlossen ist?

Die URL lässt sich nicht öffnen?

Bist du sicher, dass du eine Wi-Fi-Verbindung mit 89grad oder 89guest hast?