KLIMADATEN

Dokumentation CDK1 – FS 2021

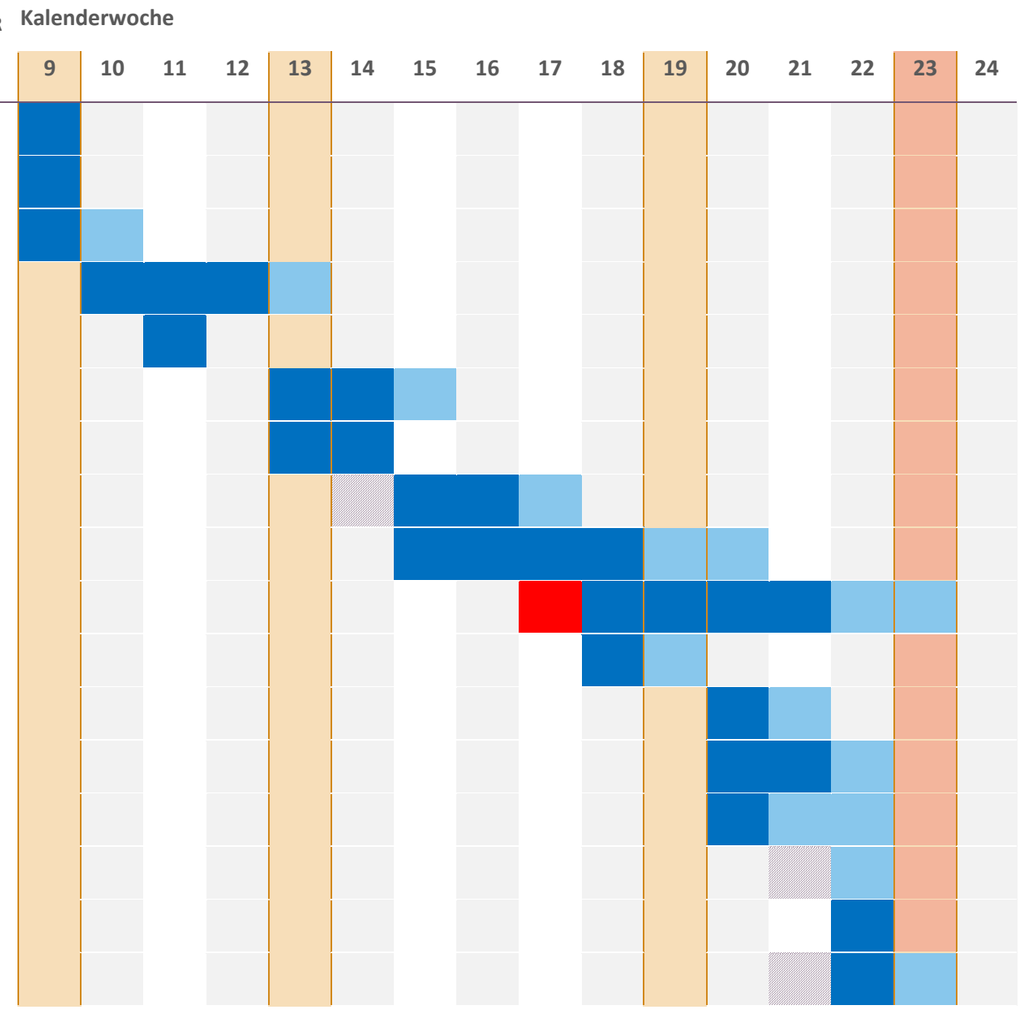
AUTOR
Julien Kellerhals, julien.kellerhals@students.fhnw.ch
Joel Grosjean, joel.grosjean@students.fhnw.ch
Lars Altschul, lars.altschul@students.fhnw.ch

CDK1 Projektplan

Es folgt eine Legende, die das Diagramm beschreibt.

geplanter Start | Umsetzung | länger als geplant

AKTIVITÄT	START DES PLANS	DAUER DES PLANS	TATSÄCHLICHER START	TATSÄCHLICHE DAUER
Thema definieren	9	1	9	1
Technologien definieren	9	1	9	1
Daten Quellen suchen	9	1	9	2
Webscraping entwickeln	10	3	10	4
Datenbank aufsetzten	11	1	11	1
API programmieren	13	2	13	3
DB Funktionen programmieren	13	2	13	2
API & ETL programmieren	14	3	15	3
Admin-Bereich entwickeln	15	4	15	6
Plots für Dashboard entwickeln	18	4	18	6
Story definieren	18	1	18	2
Quellen suchen	20	1	20	2
Dashboard finalisieren	20	1	20	3
Story schreiben	20	1	20	3
Plots für Story	21	1	22	1
Dokumentation finalisieren	22	1	22	1
Unittests schreiben	21	2	22	2



Projektplanung

In-Flight Workshop 1 – KW 13

- Thema definieren
    - Wir haben uns für extreme Wetterereignisse und Umweltkatastrophen entschieden. Im weiteren Verlauf haben wir entschieden, uns auf die Ursache für Massenbewegungen zu konzentrieren
- Technologien
    - Die genaue Auflistung der Technologien folgt in der Technischen Beschreibung
- Datenquellen suchen
    - Um die Ursachen von Massenbewegungen analysieren zu können, benötigen wir insbesondere die Daten von Idaweb.ch. Weiter haben wir noch die homogene Messreihe von MeteoSchweiz verwendet.
- Webscraping entwickeln
    - Um die Daten automatisch von unseren Quellen zu laden haben wir ein Webscraping mittels Selenium entwickelt.
- Datenbank aufsetzen
    - Wir haben uns dafür entschieden eine PostgreSQL Datenbank für unser Projekt zu verwenden. Diese können wir mit SQL Alchemy automatisch anlegen.

In-Flight Workshop 2 – KW 19

- API & ETL programmieren
    - Die API mittels Flask implementieren. Einen ETL um die Daten von Webscraping die im Stage-Schema sind ins Core-Schema zu laden.
- Admin-Bereich entwickeln
    - Admin-Bereich, um das Webscraping zu starten, den ETL Prozess zu starten und einen überblick über die Datenbank zu haben.
- Plots für Dashboard entwickeln
    - Passende Plots für unser Dashboard suchen und ein Mock-up entwerfen. Anschliessend die Plots mit Plotly und Dash implementieren.

- Story definieren
    - Definieren was die Story genau aussagen soll. Was ist die Zielgruppe und welches Narrative-Muster soll verwendet werden.

Finale Abgabe

- Quellen suchen
    - Passende Quellen für die Story suchen. Interview mit betroffenen Personen halten
- Dashboard finalisieren
    - Dashboard im Unterricht zeigen. Verbesserungen am Dashboard vornehmen.
- Story schreiben
    - Finale Story gliedern und schreiben. Plots für die Story entwickeln
- Dokumentation finalisieren
    - Dokumentation für die Abgabe überprüfen und ergänzen
- Unittest schreiben
    - Unittest mit Pytest für Webscraping und db.py schreiben.

Arbeitsjournal

KW9 (1. – 7. März 2021)

Julien - API: Habe die 3 Projekte initialisiert (Nur eins wurde dann benötigt, da das FE und die DB nicht getrennt sind). Ich habe ebenfalls einige flask-routes hinzugefügt.

KW10 (8. – 14. März 2021)

Julien - DB: Habe die Programmier-Umgebung aufgesetzt und Datenbank Verbindung aufgebaut.

KW11 (15. – 21. März 2021)

Julien - API: Automatischer Browser Treiber herunterladen code geschrieben (Überprüft die Version und den Typ).

Lars – DB: DB-Funktionen programmiert. Anlegen einer neuen Datenbank und einer Tabelle

Julien - IDAWEB: Habe das config-File für das automatische Daten Herunterladen von IDAWEB angefangen.

Joël – Webscraping: Erstellte die Basis der Navigation des Webscraping. Webscraping kann bis anhin nur durch die Blitz-Daten navigieren.

KW12 (22. – 28. März 2021)

Julien - IDAWEB: Anpassungen für die Navigation auf IDAWEB vorgenommen. IDAWEB-Daten sind in der Tabelle nicht unique, was zur Folge hat, dass wir einige Einträge nicht herunterladen. Ich werde demnächst ein Scraping die verfügbaren Daten auf IDAWEB implementieren und versuchen die Daten mittels POST-Request auszuwählen.

Julien – API: Ein Prototyp für Server-Sent-Events angefangen, so können wir die Admin-Seite dynamisch und ohne refresh aktualisieren.

Joël - IDAWEB: Leider wurde unser IDAWEB-Account wegen zu vielen Anfragen gesperrt, wir müssen auf neue Login-Daten warten. Ich habe einige Parameter herausgesucht, welche wir später herunterzuladen versuchen werden.

Julien - API: Flaks-Server unterstützt leider nur teilweise multi-threading. Die Abfragen auf den Server werden erst nach der Fertigstellung der vorherigen Abfrage angefangen. Ich habe viel Zeit investiert, um den Server teilweise multi-threaded zu machen. Leider gehen die Rückgaben nicht mehr.

Julien – FE: Integriere die Tests auf der Webseite, muss leider auch in einem Sub-Prozess ausgeführt werden, weil sonst der Server blockiert wird. Hat jetzt zur Folge das andere Prozesse vor der Fertigstellung der Tests aufhören.

Julien - API: Noch einige Änderungen am Selenium-Web-Driver vorgenommen da Chrome und Edge nicht die gleichen Treiber benutzen und die Seiten nicht vollständig geladen werden.

Lars – DB: Die Funktionen zur Erstellung einer Datenbank in eine Klasse umgewandelt.

Lars – Tests: Erster Pytest geschrieben, um das Webscraping von MeteoSchweiz zu testen. Die ersten Tests, welche noch als Funktionen programmiert wurden, habe ich in eine Klasse umgewandelt.

KW13 (29. März – 4. April 2021)

Julien - Webscraping: Einige Änderungen an der Konfiguration des Webscrapings vorgenommen.

Julien - FE: Habe die Admin Seite angefangen, mit Web-Treiber-Status, DB-Treiber, Navbar und einige Placeholder-Sites. SSE wird jetzt für die FE-sync eingesetzt und wurde auf der API implementiert (Ist leider nicht sehr flexibel).

Julien - FE: Die Zustände der Treiber auf der Admin Seite werden jetzt endlich auch korrekt aktualisiert bei Abstürzen und anderem.

Julien – API: Habe unser langes app.py in mehrere sub-APIs getrennt, da es nicht mehr lesbar und zu lange war. Ich habe dafür die Flask-Blueprints verwendet.

Julien - FE: Aktionen wie DB erstellen oder Tabelle anlegen können jetzt mittels Knopfdruck auf der Admin Seite gemacht werden.

Joël - Webscraping: Haben neue Login-Daten erhalten.

KW14 (5. – 11. April 2021)

--

KW15 (12. – 18. April 2021)

Joël - Webscraping: Da die IDAWEB-Datenanfragen zu gross sind, müssen wir diese Teilweise unterteilen. Ich habe versucht dies durch das Unterteilen in verschiedene Zeitabschnitte zu erreichen, dies funktionierte aber nicht, weil die meisten Parameter in dieselben Zeitabschnitte unterteilt werden und somit keine Unterselektion erreicht werden kann. Deshalb habe ich nun die Datenanfragen in Abschnitte gemäss Höhe der Station unterteilt. Wenn also zu viele Daten in einer Anfrage sind, wird der Höhenabschnitt eingeschränkt, bis die Datenmenge genug klein ist. Dies funktioniert erstaunlich gut.

Julien – API: Daten werden von dem Stage-Schema in das Core-Schema geladen. Dies dient der Versionierung und Homogenisierung der Daten.

Julien - FE: Materialized ist jetzt auf die Admin Seite implementiert. Dies vereinfacht es die Material-Design-Richtlinien von Google einzuhalten.

Julien - Webscraping: Habe Logik für das Herunterladen der Bestellungen von IDAWEB geschrieben.

Julien - FE: Implementiere die Breadcrumb-navbar auf der Tabellen-Seite, Idee ist der Zustand der Datenbank abzubilden und die richtigen Aktionen dann auszuführen (DB erstellen, Tabelle anlegen). Dies ersetzt die momentanen Buttons. Die Buttons der Tabellen werden bei der Aktualisierung der Daten jetzt deaktiviert.

Joël - Dashboard: Habe angefangen das Dashboard zu erstellen. Die Verbindung zu zwischen Flask und Dash stellt sich als komplizierter heraus als gedacht.

Julien - Dashboard: Haben die Dash-App mit dem bestehenden Flask-Server verbunden. Es hat sich gezeigt, dass dies relativ komplex ist. In der ersten Phase tritt der Request auf die Flask-API zu, die wird dann als nächstes weitergeleitet zu einem anderen Flask Server (Dash-Apps laufen ebenfalls über eine Flask-App). Der Dash-Server verarbeitet dann die Abfrage und gibt

dem Browser die Seite zurück. Somit besteht am Ende eine direkte Verbindung zwischen dem Browser und die Dashapp?? (wcgw)

Lars – Test: Weitere Tests geschrieben für das Webscraping.

KW16 (19. – 25. April 2021)

Julien - FE: Auf der Admin Seite sieht man jetzt wie viele Zeilen in die Tabelle geladen wurden und wann dies getan wurde. Die JS-Scrips sind in getrennten Files gespeichert.

Julien - DB: Die Daten werden jetzt ohne Duplikate ins Core geladen. Ein Index wurde der Tabelle hinzugefügt, weil die Tabelle sehr langsam wurde. Jetzt werden auch noch die IDAWEB Station und Parameter Informationen geparsed und in eine tabellarische Form gebracht und bis zum Core geladen. Die IDAWEB-Tabelle kann in mehreren Teilen geladen werden. Es werden nur die fehlenden oder neuen Daten geladen.

KW17 (26. April – 2. Mai 2021)

Joël - Dashboard: Habe angefangen das Dashboard zu erstellen. Beim Testen merkte ich, dass die Callbacks nicht funktionierten. Deshalb musste ich Dash anders einbinden, um die Callbacks zu ermöglichen. Habe an der grundsätzlichen Struktur und dem Layout des Dashboards gearbeitet. Weiterhin habe ich einen ersten Plot mit Callback erstellt. Habe mit dem formatieren des Plots angefangen. Habe die Mathematik für das Errechnen von Regressionslinien hinzugefügt. Habe eine noch nicht schöne Landkarte hinzugefügt, welche die Stationen als Punkte darstellt. Es scheint schwierig zu sein, die Landkarte zu verschönern. Ich versuchte es mit einem Geojson, es funktionierte jedoch nicht. Habe den Link zur Datenstory erstellt. Habe einen Scatterplot für den Schneefall mit Callback hinzugefügt.

Julien - FE: Habe die ganzen Spalten im Front-End dynamisch gemacht, jetzt muss die Struktur nur noch in der API definiert werden, und das Front-End kümmert sich um das Anzeigen der Zeilen und Spalten. Dies hat den Vorteil das es nur einmal implementiert werden muss und über alle Seiten der Admin Page verwendet werden kann. Die Backend-SSE wurden auf die neue FE-Struktur angepasst. Jetzt werden Server-Status, Stage- und Core-Tabellen über die neue SSE-Implementierung angepasst.

Julien - DB: Habe DB-Triggers auf alle Tabellen der Datenbank hinzugefügt, count_nrow und last_refresh mv erstellt. Die werden dank den neuen Triggers nach jedem Update aktualisiert. Dies hat zur Folge das Pandas nur Row-by-Row-Inserts macht.

KW18 (3. – 9. Mai 2021)

Joël - Dashboard: Ich habe die Punkte auf der Landkarte entfernt, welche zu inkonsistente Daten enthielten. Habe einen weiteren Scatterplot für den extremen Regen hinzugefügt. Habe einen Fehler mit den Koordinaten festgestellt und diesen Behoben. Die Zahlen hinter dem Komma bei den long- und lat-Koordinaten wurden teilweise falsch gerundet. Habe ebenfalls ein Problem mit der Regressionslinie gefunden: Der Index musste zurückgesetzt werden um Fehler zu vermeiden. Habe ebenfalls die Scatterplots etwas besser formatiert.

Lars – Datenstory: Quellen für die Story suchen. Das Konzept für die Story mit narrativem Muster, Zielgruppe etc. erstellen.

KW19 (10. – 16 Mai 2021)

Joël - Dashboard: Habe aus dem Plot für den extremen Regen einen Bar-Chart gemacht, da dieser die Abweichung vom Durchschnitt etwas besser aufzeigt als ein Line-Chart. Habe ebenfalls noch mehr Stationen von der Landkarte entfernt, da es immer noch viel zu viele Stationen mit schlechten Daten und gigantischen Datenlöchern gab. Deshalb ist das neue Kriterium, um eine Station auf der Landkarte anzuzeigen, Messungen von mindestens 43 Jahren zu haben und zwischen dem minimalen und maximalen Datum mehr als 95% der Jahre zu haben. Dies verhindert grössere Datenlöcher sehr gut. Ich habe ebenfalls den Plot für den Schneefall für alle Stationen hinzugefügt. Ich habe ebenfalls herausgefunden, dass man eine schöne Landkarte mithilfe einer Scattermapbox von OpenStreetMap erstellen kann. Dies habe ich auch getan und ich habe meinen eigenen Kartenstil darübergelegt. Mit Dash selbst liessen sich die Ecken der Scattermapbox nicht ausrunden. Deshalb musste ich die borders.css Datei im assets Ordner hinzufügen. Ich habe auch das Problem behoben, dass einige Stationen auf der Landkarte doppelt angezeigt wurden.

Julien - FE: Das Front-End sendet mehrmals die gleiche Abfrage zum Server. Die API wird dabei beim Abbauen der Warteschlange sehr langsam. Click Event wurde gewechselt, um dieses Problem zu beheben.

Julien - API: Das informieren des Front-Ends wird jetzt mithilfe einer helper-class durchgeführt. So befindet sich der Code an einem Ort und kann mehrmals verwendet werden.

Lars – Datenstory: Eine passende Geschichte für das narrative Muster «Humans behind the dots» finden. In einer Doku von SRF habe ich eine Geschichte von Bondo gefunden. Weiter habe ich eine passende Beitrag gefunden, dass die ETH aktuell an einem Frühwarnsystem für Murgänge mit Hilfe einer KI arbeitet.

KW20 (17. – 23. Mai 2021)

--

KW21 (24. – 30 Mai 2021)

Julien - FE: Row count und Last refresh wurden mit dem neuen Announcer implementiert und die action-buttons auf der FE-Seite wieder deaktiviert.

Joël - Dashboard: Ich habe es ermöglicht auch jährliche Daten hinzuzufügen. Zudem habe ich mich entschieden die Callbacks so zu gestalten, dass ein ausgewählter Punkt auf der Karte alle umliegenden Plots anpasst und dort die ausgewählte Station anzeigt. Um dies zu ermöglichen musste ich die Stationen weiter filtern. Ich habe sie so gefiltert, dass ich eine hohe Datenkonsistenz bei allen Scatterplots und Barcharts garantieren konnte. Deshalb ist das neue Kriterium um eine Station auf der Landkarte anzuzeigen Messungen bei allen benutzten Parametern von mindestens 30 Jahren zu haben und zwischen dem minimalen und maximalen Datum mehr als 90% der Jahre zu haben. Dies verhindert grössere Datenlöcher sehr gut und lässt etwa 20 Stationen zum Auswählen übrig. Da das Data-Wrangling etwas unübersichtlich wurde, habe ich es in einzelne allgemeine Funktionen unterteilt.

Lars – Datenstory: Kontakt mit dem WSL aufgenommen, um Daten über Murgänge und deren Folgen zu bekommen. Sie geben keine Daten heraus. Nur bereits publizierte Auswertungen

der Daten welche für unsere Story nicht passend sind. Habe die Kontaktdaten der Familie Buob-Müller erhalten. Mit ihnen habe ich einen Termin für ein Interview ausgemacht, um weiter Informationen für unsere Story zu erhalten.

KW22 (31. Mai – 6. Juni 2021)

Joël - API: Ich habe PATCH und DELETE als Teil der CRUD-Fähigkeiten der API hinzugefügt. Diese Änderungen dienen der Abgabe von wdb und nicht direkt dem Projekt selbst.

Julien - API: Postgres-connection-string-Seite wurde geschrieben und im bestehenden code implementiert.

Joël - Dashboard: Ich habe das Data-Wrangling für das Dashboard übersichtlicher gestaltet und einige kleine Probleme behoben. Zudem habe ich einen Plot für den Regen mit einem Callback erstellt. Ich habe auch einen Button erstellt, mit dem sich alle Stationen anzeigen lassen und habe das Callback dafür erstellt. Ich habe auch eine Beschreibung zum Dashboard hinzugefügt, welche auch einen Teil enthält, welcher darüber Auskunft gibt, welche Station gerade angezeigt wird. Da diese Beschreibung je nach Zoom zu gross wird, musste ich dafür eine Scrollbar benutzen. Diese Scrollbar lässt sich nicht direkt mit Dash modifizieren, weshalb ich die Datei style.css im Ordner assets erstellt habe. Ich hatte die Idee, dass eine Heatmap die extremen Regefälle evtl. besser darstellen könnte. Ich habe diese Idee nach dem Erstellen der Heatmap jedoch verworfen, weil die Änderungen innerhalb eines Jahres viel grösser sind als die Änderung über die Jahre hinweg und man somit keine Veränderung feststellen kann.

Joël - Datenstory: Ich habe angefangen das grobe Layout der Datenstory zu erstellen. Ich habe dafür unter anderem einen Header und einen Test-Plot hinzugefügt.

Joël - Webscraping: Ich habe einige obsolete Funktionen aus dem Webscraping entfernt.

Joël – Unit-Tests: Ich habe einige Unit-Tests für die Datenbank hinzugefügt. Diese dienen dem Modul wdb und haben keinen Einfluss auf die Funktionalität des Projekts.

Lars – Tests: Weitere Unittests geschrieben, um die Funktionen im Webscraping von IDAWEB zu testen.

Lars – Datenstory: Interview mit Reto Müller geführt und die Story fertig geschrieben.

KW23 (7. – 13. Juni 2021)

Julien: Habe das Readme geschrieben.

Joël: Habe das Readme verbessert.

Joël: Habe versucht das requirements.txt so zu erstellen, dass es möglich ist das Projekt auf einem neuen Environment zu starten und nur das requirements.txt auszuführen um die nötigen Libraries und zu installieren. Habe jedoch nach einigen Versuchen aufgegeben, da ich nicht herausfand wie dies gemacht werden muss und zusätzlich über den Nutzen dieser Arbeit zweifelte.

Lars – Webscraping: Webscraping von MeteoSchweiz überarbeitet. Die einzelnen Schritte in mehrere Funktionen unterteilt.

Julien – Webscraping: Habe das IDAWEB Login Problem gelöst, bei dem man sich bei bestehender Session zwei Mal anmelden muss.

Joël – Datenstory: Habe das Data Wrangling für die Datenstory-Plots angepasst und fertiggestellt. Habe den Text der Datenstory überarbeitet und das Interview in einen Lauftext umgewandelt. Habe auch die Datenstory visuell etwas überarbeitet und fertiggestellt.

Julien - DB: Implementierung der neusten SQLalchemy-Version.

KW24 (14. –20. Juni 2021)

Julien: Habe am Readme geschrieben.

Julien: Habe die Verbindung zwischen der Flask-App und dem Dash Server verbessert. Jetzt kann man den Server ohne auskommentieren der Dash-App beim ersten aufstarten des Projekts ausführen.