HuggingEarth:
Web Catalogue for User-Friendly Search and Retrieval of Machine Learning Models for EO Datacubes

TerraLink

28. Oktober 2024

Projektbezeichnung	Web Catalogue for Machine Learning Models for EO Datacubes
Projektleitung	TerraLink
Erstellt am	18. Oktober 2024
Letzte Änderung am	28. Oktober 2024
Status	In Bearbeitung
Aktuelle Version	1.0

Inhaltsverzeichnis

1 Einleitung

1.1 Projektübersicht

Das Ziel dieses Projekts ist die Entwicklung von **HuggingEarth**, einem benutzerfreundlichen Web-Katalog, der die Suche und den Abruf von maschinellen Lernmodellen für Earth Observation (EO) ermöglicht. Der Katalog basiert auf der SpatioTemporal Asset Catalog (STAC)-Spezifikation und der STAC Machine Learning Model (MLM)-Erweiterung, um die Integration von ML-Modellen in Workflows der Nutzer zu unterstützen.

1.2 Über TerraLink

TerraLink ist ein führendes Unternehmen im Bereich der Entwicklung und Bereitstellung von Plattformen für maschinelles Lernen, spezialisiert auf Anwendungen für Erdbeobachtungsdaten. Unsere Mission ist es, fortschrittliche Technologien und benutzerfreundliche Lösungen für die Verwaltung und Verarbeitung von spatio-temporalen Daten zu liefern.

1.3 Zweck des Dokuments

Dieses Pflichtenheft definiert die Anforderungen und Rahmenbedingungen für das zu entwickelnde System. Hiermit stellen wir eine verbindliche Vereinbarung zwischen Ihnen und unserem Team TerraLink bereit und beschreiben unsere geplante Realisierung der von Ihnen genannten Anforderungen.

2 Anforderungen aus dem Lastenheft

Folgende Anforderungen aus dem Lastenheft werden im Detail umgesetzt:

- **Katalogisierung von ML-Modellen:** Bereitstellung eines durchsuchbaren, filterbaren Katalogs von ML-Modellen für die Erdbeobachtung, mit Metadaten gemäß der STAC MLM-Erweiterung.
- **Modell Integration:** Ermöglicht es den Nutzern, diese Modelle über PyStac in ihre Workflows zu integrieren.
- **Upload und Download von STAC-konformen Metadaten:** Unterstützung des Hoch- und Herunterladens von raum-zeitlichen Metadaten, so dass die Nutzer Modelle in Standardformaten abrufen können, die sich leicht in ihre Python- oder R-Workflows integrieren lassen.
- **Community Contribution:** Benutzer können ihre eigenen Modelle hochladen und somit einen Beitrag zum Repository der Plattform leisten, das der STAC MLM-Erweiterung folgt.

3 Lieferumfang

Wir liefern eine komplette Softwarelösung, die den Anforderungen aus dem Lastenheft entspricht und wie folgt aufgebaut ist:

- Eine **Homepage**, die die Benutzer über die Plattform informiert und den Zugang zu allen Funktionen bietet.
- Eine **Unterseite zur Suche, Filterung, Anzeige und Download von Modellen.** Über ein Suchfeld kann man nach Stichworten suchen. Anschließend kann man die zur Suche passenden Modelle filtern. Wenn man ein Modell ausgewählt hat werden detaillierte Informationen zu den Modellen angezeigt. Außerdem werden visuelle Vorschauen bereitgestellt, um den Benutzern zu helfen, die Eignung eines Modells zu bewerten. Anschließend können die STAC-konformen Metadaten und die Modelldateien über einen Button heruntergeladen werden.
- Eine **Unterseite für den Upload von Modellen**, die sicherstellt, dass Benutzer eigene Modelle und Metadaten gemäß der STAC-MLM-Spezifikation hochladen können. Falls diese nicht eingehalten werden, erscheint eine Fehlermeldung auf der Website, die genauere Informationen enthält.
- Eine **Unterseite mit Tutorials** zur Nutzung der Plattform, einschließlich häufig gestellter Fragen (FAQs).
- Eine **Unterseite mit Tutorials zu RSTAC und PySTAC**, die detailliert erklärt, wie man die Modelle abfragt, anwendet und direkt in Python- oder R-Workflows integriert. Hierzu werden Code-Snippets bereitgestellt und eine Schritt-für-Schritt-Anleitung geliefert.
- Ein **Impressum** gemäß den rechtlichen Anforderungen.
- (Bonus: Eine **Unterseite zur Verwaltung des Nutzerkontos**, um die Anmeldung, das Profilmanagement und die Verwaltung der hochgeladenen Modelle zu ermöglichen.)

4 Produktfunktionen und Ziele

4.1 /F10/ (/LF10/): Modellkatalog und Suche

Produktfunktion: Bereitstellung eines durchsuchbaren, filterbaren Katalogs für ML-Modelle, der auf der STAC-MLM-Erweiterung basiert und eine intuitive Benutzeroberfläche bietet.

Kernfunktionalitäten:

- **STAC-konforme Modell-Metadaten**:
  - **Datenmodell**: Ein Datenmodell wird entwickelt, das alle Metadatenfelder der STAC-MLM-Erweiterung abdeckt, einschließlich:
    - **Model task**: Attribut mit vordefinierten Werten wie "classification", "regression", "anomaly detection".
    - **Temporal and spatial applicability**: Felder für Start- und Enddatum (temporal), Geometrien für räumliche Daten (z.B. Region of Interest als GeoJSON).
    - **Data type**: Attribut, das auf verfügbare Datentypen verweist.
    - **Input/output requirements**: Attribut für Anforderungen an Eingabe- und Ausgabedaten (z.B. Auflösung, Bänderanzahl, Dateiformat).
    - **Validierung**: Implementierung einer Validierungslogik, die sicherstellt, dass alle Metadaten den STAC-MLM-Richtlinien entsprechen, bevor sie gespeichert werden können.

2. Such- und Filtermöglichkeiten:

- **Suchfunktion**: Implementierung einer Such-API, die Anfragen basierend auf den folgenden Parametern zulässt:
  - **Model task**: Suchfilter basierend auf vordefinierten Aufgaben (z.B. "classification").
  - **Geografische Region**: Verwendung von Geodaten (GeoJSON) zur Filterung nach räumlicher Relevanz.
  - **Data type**: Filterung basierend auf unterstützten Datentypen.
  - **Zeitlicher Bereich**: Filterung nach Start- und Enddatum.
  - **Cloud Coverage**: Filterung nach der prozentualen Wolkenabdeckung.

3. Modell-Exploration:

- **Metadatenseite pro Modell**: Entwicklung einer Ansicht, die für jedes Modell detaillierte Informationen anzeigt. Diese Seite enthält:
  - **Modellbeschreibung**: Ein Textfeld, das den Zweck des Modells und die zugrunde liegende Architektur erklärt.
  - **Anwendungsfälle**: Angabe der empfohlenen Anwendungsfälle und potenziellen Einschränkungen.
  - **Hyperparameter**: Darstellung der für das Modell verwendeten Hyperparameter.

* **Input/Output-Formate**: Auflistung der Eingabe- und Ausgabeformate des Modells.
* **Zusätzliche Anmerkungen**: Weiterführende Hinweise zur Modellnutzung.

Implementierungsstrategie: Die Such- und Filterlogik wird über eine API-Endpunktstruktur im Back-End abgebildet, die Suchanfragen über MongoDB abwickelt. Indizierung der wichtigsten Filterfelder stellt eine performante Verarbeitung sicher. Eine kombinierte Abfrage ermöglicht es Nutzern, mehrere Filterparameter zu kombinieren, um präzise und relevante Suchergebnisse zu erzielen. Ergebnisse werden dem Nutzer in einer Ergebnisliste mit Seitennummerierung zurückgegeben, die die wesentlichen Metadaten des Modells enthält und nach Relevanz sortiert ist.

4.1.1 Anwendungsfall

Suche nach Modellen für die Waldüberwachung

Beschreibung: Ein Umweltforscher möchte mithilfe von Erdbeobachtungsdaten die Veränderung und den Zustand von Wäldern überwachen. Ziel ist es, ein Machine-Learning-Modell zu finden, das sich für die Klassifikation und Detektion von Waldgebieten eignet und Veränderungen wie Abholzung oder Waldbrände identifizieren kann.

Schritte im System:

1. Der Benutzer öffnet die Suchseite und gibt im Suchfeld „Waldüberwachung“ ein.
2. Im Filterbereich wählt der Benutzer:
   - **Modellaufgabe**: Klassifikation oder Anomaliedetektion (um Modelle zu finden, die auf die Erkennung von Veränderungen spezialisiert sind).
   - **Datentyp**: Sentinel-2 (geeignet für Vegetationsanalyse) oder Landsat (ermöglicht zeitliche Vergleichbarkeit über lange Zeiträume).
   - **Geografische Region**: Der Benutzer gibt die gewünschte Region ein, z. B. ein Land oder eine spezifische Region wie den Amazonas.
   - **Zeitspanne**: Der Benutzer definiert eine Zeitspanne, um nur Modelle anzuzeigen, die Daten in dieser Periode verarbeiten können.
   - **Cloud Coverage**: Der Benutzer setzt eine maximale Wolkenabdeckung von 10 %, um Modelle auszuwählen, die klare Bilder voraussetzen.
3. Das System führt eine Abfrage in der MongoDB-Datenbank durch und zeigt eine Liste der Modelle an, die den festgelegten Kriterien entsprechen.
4. Der Benutzer wählt ein Modell aus der Liste aus und öffnet die Detailansicht. Diese Ansicht enthält:
   - Eine Modellbeschreibung, die den Anwendungsfall „Waldüberwachung“ bestätigt.
   - Informationen zur Architektur und spezifischen Anwendungsfällen.
   - Vorschauen der Eingabedaten, die die Modellabdeckung und typische Ergebnisse für Waldgebiete visualisieren.

5. Der Benutzer kann das Modell über einen Button herunterladen oder die STAC-konformen Metadaten einsehen und direkt in seinen Workflow einbinden.

Erwartetes Ergebnis: Der Forscher findet ein Modell, das speziell auf die Waldüberwachung ausgerichtet ist, und erhält alle notwendigen Informationen, um das Modell für die Analyse und kontinuierliche Überwachung von Waldgebieten einzusetzen.

4.2 /F20/ (/LF20/): Modell-Upload und Metadatenverwaltung

Produktfunktion: Die Upload- und Verwaltungsfunktionalität ermöglicht Benutzern, neue Modelle mit STAC-konformen Metadaten in den Katalog hochzuladen. Der Prozess stellt sicher, dass die Modelle den Anforderungen der STAC-MLM-Spezifikationen entsprechen und für die Abfrage und Integration durch STAC-kompatible Clients bereitstehen.

Der Upload-Prozess erfolgt in mehreren Schritten, die jeweils eine Validierung und Benutzerunterstützung bieten:

- **Schritt 1: Metadatenerfassung**
  Der Benutzer gibt grundlegende Modellinformationen an, darunter:
  - **Modellname und -beschreibung**: Klarer Name und Zweck des Modells.
  - **Modellaufgabe**: Auswahl der Aufgabenart (z.B. Klassifikation, Regression, Anomaliedetektion) aus vordefinierten Optionen.
  - **Ein- und Ausgabeanforderungen**: Angaben zu Datentyp, Auflösung, Anzahl der Bänder und Dateiformat.
- **Schritt 2: Angaben zur Datenquelle und Trainingsdetails**
  Die Benutzer geben die verwendeten Erdbeobachtungsdaten an und spezifizieren die räumlichen und zeitlichen Parameter des Trainingsdatensatzes. Wichtige Angaben sind:
  - **Datenquelle**: Typ des Datensatzes (z.B. Sentinel-2).
  - **Räumliche und zeitliche Abdeckung**: Angabe der geografischen Abdeckung im GeoJSON-Format und des Zeitbereichs im ISO 8601-Format (YYYY-MM-DD).
- **Schritt 3: Modell-Upload**
  Benutzer laden die Modelldateien hoch, wobei folgende Formate unterstützt werden:
  - Unterstützte Dateiformate: ONNX (.onnx), PyTorch (.pt), TensorFlow (.h5).
- **Schritt 4: Metadatenvalidierung und Fehlerbehebung**
  Während des Uploads werden die Metadaten auf Vollständigkeit und Konformität mit den STAC-MLM-Spezifikationen geprüft:

    - **STAC-Konformität**: Überprüfung, ob alle erforderlichen Felder vorhanden und korrekt formatiert sind. Die räumliche Abdeckung wird im GeoJSON-Format und die zeitlichen Daten im ISO 8601-Format erwartet.
    - **Datenintegrität**: Sicherstellung, dass die geografische und zeitliche Abdeckung den Angaben in den Trainingsdetails entspricht.
    - **Fehlermeldungen**: Bei Problemen werden präzise Fehlermeldungen angezeigt, z. B. „Das Feld ‚Model Task' ist erforderlich und muss einen vordefinierten Wert enthalten" oder „Die geografische Abdeckung muss im gültigen GeoJSON-Format vorliegen".
- **Schritt 5: STAC-Metadatengenerierung**
  Das System generiert automatisch ein STAC-kompatibles Metadatendokument, das für STAC-Clients abrufbar ist. Die Validierung stellt sicher, dass:
    - Alle wesentlichen Felder ausgefüllt sind und die STAC-MLM-Spezifikationen erfüllen.
    - Die Struktur der Metadaten für die Abfrage und Integration mit PySTAC und RSTAC optimiert ist.
- **Schritt 6: Bestätigung und Benachrichtigung**
  Nach erfolgreichem Upload erhält der Benutzer eine Erfolgsmeldung. Das Modell kann im Benutzerprofil unter "Meine Modelleängesehen und verwaltet werden und ist im Suchkatalog auffindbar.

Benutzerunterstützung und Anleitung:
Die Plattform bietet eine interaktive Schritt-für-Schritt-Anleitung. Hilfetexte und Tooltipps unterstützen den Benutzer bei der korrekten Eingabe der Daten und liefern Beispiele für Metadaten im STAC-MLM-Format.

Modellstruktur für MongoDB
Die hochgeladenen Modelle und Metadaten werden in MongoDB gespeichert, wobei Mongoose für Schema-Validierungen und Fehlerbehebung integriert ist. Die folgenden Schema-Definitionen sind enthalten:

- **Modell-Schema (ModelSchema)**: Speichert zentrale Modellmetadaten wie:
    - **name** (Modellname), **description** (Beschreibung), **modelTask** (z.B. Klassifikation).
    - **temporalCoverage** und **spatialCoverage** (zeitliche und räumliche Abdeckung).
    - **dataType** (z.B. Sentinel-2), **inputRequirements** und **outputRequirements** (Ein- und Ausgabeanforderungen).
    - **filePath** (Pfad zur Modell-Datei), **uploadedBy** (Benutzer-ID des Uploaders).
- **Benutzer-Schema (UserSchema)**: Speichert Benutzerdetails, z. B.:
    - **username** (Benutzername), **email** (E-Mail-Adresse), **role** (Benutzerrolle, z.B. „admin", „user").

– **Upload-Log-Schema (UploadLogSchema)**: Verfolgt den Status jedes Uploads (z.B. „pending“, „approved“ oder „rejected“) und listet Validierungsfehler auf.

Diese Struktur gewährleistet, dass alle wesentlichen Metadatenfelder der STAC-MLM-Spezifikationen entsprechen und effizient in der MongoDB-Datenbank gespeichert werden. Das Back-End verwaltet die Dateneingabe und Validierung dieser Informationen mithilfe von Mongoose.

4.3 /F30/ (/LF30/): Integration mit STAC-Clients

Produktfunktion: Unterstützung von STAC-Clients wie PySTAC und Bereitstellung von Modellen und Metadaten für direkte Integration.

Kernfunktionalitäten:

– **STAC-Client Support:** Die Anwendung unterstützt den Abruf von Modellen und Metadaten via PySTAC-Client. Zudem werden Anleitungen und Code-Snippets bereitgestellt, um Modelle und Metadaten in Python-Workflows zu integrieren.

– **Model Fetching:** Die Anwendung stellt die STAC-Metadaten für jedes maschinelle Lernmodell zum Download bereit. Die Metadaten enthalten Links zu den Modell-Dateien, die User über STAC-kompatible Clients herunterladen oder abrufen können.

– **Demonstration der Client-Integration:** Die Integration in PySTAC wird so erweitert, dass User Modelle über das Catalog-Frontend suchen und abrufen können.

4.4 /F40/ (/LF40/): Performance

Reaktionszeit von 1 Sekunde für Benutzerinteraktionen

– **Strategie**: Verwendung von Asynchronität und Lazy Loading, um die Ladezeiten für kritische Elemente zu minimieren. Daten werden über AJAX geladen, sodass die Seite ohne vollständiges Neuladen aktualisiert werden kann.

– **Frontend-Optimierung**: Einsatz von Code-Splitting und Caching, um die Ladezeiten zu minimieren und die Benutzeroberfläche reaktiv zu halten.

Reaktion innerhalb von 0,1 Sekunden für interaktive Visualisierungen

– **Strategie**: Nutzung von WebGL und D3.js für hochperformante interaktive Grafiken. WebGL bietet Hardware-Beschleunigung und ermöglicht schnelle Rendering-Zeiten, ideal für Karten und Modelle.

– **Serverseitiges Caching**: Häufig abgerufene Metadaten und Modelle werden im Cache gespeichert, um die Antwortzeiten zu verkürzen.

4.5 /F50/ (/LF50/): Sicherheit

Zur Sicherstellung der Datensicherheit und Integrität werden folgende Sicherheitsanforderungen implementiert:

- **Datenverschlüsselung**: Alle Datenübertragungen müssen SSL-verschlüsselt sein.
- **Zugriffskontrolle**: Nur autorisierte Nutzer dürfen auf bestimmte Funktionen (z.B. Modell-Upload) zugreifen. Ein Rollen- und Rechteverwaltungssystem regelt den Zugriff.
- **Datenintegrität**: Nutzerdaten und Modellmetadaten dürfen nur von authentifizierten und autorisierten Nutzern geändert werden.
- **Audit Logging**: Alle sicherheitsrelevanten Aktivitäten sollen geloggt und regelmäßig auf Anomalien überprüft werden.

4.6 /F60/ (/LF60/): Wartbarkeit

Open Source und Lizenzkompatibilität

- **Strategie**: Das Projekt wird unter einer OSI-konformen Open-Source-Lizenz veröffentlicht. Alle externen Bibliotheken und Tools werden überprüft, um sicherzustellen, dass deren Lizenzen mit der gewählten Lizenz kompatibel sind.

Dokumentation

- **Strategie**: Alle Schritte zur Entwicklung, Installation, Konfiguration und Testung der Plattform werden in einer strukturierten Dokumentation festgehalten. Markdown-Dokumente und ein GitHub Repository dienen als Plattform für die Online-Dokumentation..

5 Produktleistungen

5.1 /L10/ (/LL10/): Daten- und Metadaten-Download

- **STAC-Metadaten-Download**: Nutzer können die STAC-Metadaten eines Modells durch Klick auf einen Button herunterladen und z.B. in PySTAC verwenden.
- **Modell-Datei-Download**: Bereitstellung von Download-Links für Modell-Dateien (z.B. ONNX, TensorFlow, PyTorch).

5.2 /L20/ (/LL20/): Community-Contribution und Verwaltung

- Ein Benutzersystem mit Registrierung und Anmeldung sorgt dafür, dass nur verifizierte Benutzer Modelle hochladen können.

- Eine dedizierte Upload-Seite im Frontend erlaubt Benutzern das Hochladen von Modellen, einer JSON-Datei mit STAC-konformen Metadaten und einer Beschreibung.
- Eine automatische Validierung prüft die JSON-Metadaten auf Einhaltung der STAC-MLM-Spezifikationen.

5.3 /L30/ (/LL30/): Bereitstellung

Bereitstellung über Docker

- **Strategie**: Bereitstellung aller Komponenten als Docker-Container, wodurch eine einfache Installation und Konfiguration auf beliebigen Servern oder Cloud-Umgebungen möglich wird.
- **Docker Compose**: Verwendung von Docker Compose für den Aufbau und das Management der Container (Frontend, Backend, Datenbank).
- **Cloud-Kompatibilität**: Bereitstellung auf Cloud-Plattformen wie AWS oder Google Cloud, um skalierbare Lösungen zu bieten.

Docker Hub Integration

- **Strategie**: Alle Docker-Images werden auf Docker Hub bereitgestellt und sind dort leicht zugänglich. Die Images werden regelmäßig aktualisiert und versioniert, sodass ältere Versionen weiterhin verfügbar sind.

5.4 /L40/ (/LL40/): Training und Demonstration

- **Dokumentation und Tutorials**: Bereitstellung von Dokumentation, die sowohl allgemeine Benutzer als auch Entwickler unterstützt. Die Dokumentation umfasst Einführung, Installation und Setup, Benutzerhandbuch und API-Dokumentation.
- **Schritt-für-Schritt-Tutorials**: Nutzeranleitungen für Modellnutzung, Metadatenabruf und Workflow-Integration, inklusive Integration von STAC-Clients. Über das Frontend zugänglich (unter „Hilfe“).

5.5 /L50/ (/LL50/): Projektmanagement

Online-Projektmanagement-Tool

- **Strategie**: Zugang zu Jira, um alle Aufgaben, Fortschritte und Meilensteine transparent zu verwalten.

6 Benutzeroberfläche

6.1 /B10/ (/LB10/): Benutzerfreundlichkeit

Intuitive Benutzeroberfläche (UI)

- **Strategie**: Die Plattform bietet eine klare, minimalistische Oberfläche, angelehnt an Hugging Face. Die Hauptfunktionen (z.B. Modellsuche, Ansicht der Metadaten und Download) sind durch klare Schaltflächen und Menüs sofort erkennbar und zugänglich.
- **Technologie & Frameworks**: Für das Frontend wird eine Kombination aus HTML, CSS und JavaScript verwendet, um eine intuitive und reaktionsschnelle Benutzeroberfläche zu entwickeln. Bootstrap wird zur Vereinfachung der Layouts und zur Sicherstellung einer konsistenten Gestaltung genutzt.
- **Browserkompatibilität**: Cross-Browser-Tests mit BrowserStack und die Einhaltung von HTML5-, CSS3- und ES6-Standards gewährleisten eine Funktionalität auf mindestens 80% der gängigen Browser.

Barrierefreiheit für Farbenblindheit

- **Strategie**: Farbkontraste werden sorgfältig gestaltet, um sicherzustellen, dass die Farben auch von Personen mit Farbsinnstörungen unterschieden werden können.
- **Farbschemata und UX-Design**: Die Einhaltung der WCAG (Web Content Accessibility Guidelines), insbesondere für Kontraste, wird gewährleistet und Bibliotheken wie Color Brewer verwendet, um geeignete Farbpaletten zu erstellen.

- **Such- und Filterfunktionen**:
  - Einfach zu bedienende Suchfunktionen und Filteroptionen nach spezifischen Modellanforderungen.
  - Filtermöglichkeiten für geographische Region, Datentyp, Zeitspanne und Wolkenabdeckung.

6.2 /B20/ (/LB20/): Visualisierung

Technische Umsetzung der Visualisierung: Die Visualisierungsfunktionalität wird durch den Einsatz von WebGL und D3.js realisiert und bietet Nutzern eine intuitive, interaktive Darstellung der Modelldaten.

- **Technologiewahl:** WebGL ermöglicht hardware-beschleunigtes Rendering für schnelle und interaktive Kartenansichten und ist ideal für die Darstellung großer Datenmengen. D3.js wird für die Manipulation und Darstellung der Modelldaten eingesetzt, um eine dynamische und anpassbare Nutzererfahrung zu gewährleisten.
- **Interaktive Kartenansicht:** Die räumliche Abdeckung der Modelle wird mithilfe einer interaktiven Karte dargestellt, wobei Marker und Polygone zur Visualisierung der Abdeckungsregionen verwendet werden. Dies erleichtert den Nutzern die geografische Einordnung der Modelle. Kartenrenderings werden in einer Auflösung von mindestens 1920x1080 Pixeln und mit einer Bildwiederholrate von mindestens 60 FPS dargestellt.

- **Datenvorschau:** Zusätzlich zur Kartenansicht werden Vorschauen der Eingabedaten des Modells angezeigt, die es Nutzern ermöglichen, die Abdeckung und Qualität der zugrunde liegenden Daten schnell zu bewerten. Modellvorschauen zeigen Rasterdaten in einer Auflösung von mindestens 512x512 Pixeln, wobei eine Standardfarbgebung zur Unterstützung farbblinder Nutzer verwendet wird.
- **Beispielkonfigurationen:** Die Visualisierung bietet verschiedene Beispielkonfigurationen, wie z.B. die Darstellung von Sentinel-2-Bildern zur Vegetationsanalyse. Dies unterstützt Nutzer bei der Interpretation der angezeigten Daten und hilft ihnen, die Eignung des Modells für spezifische Anwendungsfälle besser einzuschätzen.

.

7 Qualitätsanforderungen

Qualitätsanforderung	**sehr gut**	**gut**	**normal**	**nicht relevant**
Funktionalität	x			
Angemessenheit		x		
Richtigkeit	x			
Interoperabilität	x			
Ordnungsmäßigkeit	x			
Sicherheit		x		
Zuverlässigkeit	x			
Reife	x			
Fehlertoleranz		x		
Wiederherstellbarkeit		x		
Benutzbarkeit	x			
Verständlichkeit	x			
Erlernbarkeit		x		
Bedienbarkeit	x			
Effizienz		x		
Zeitverhalten		x		
Verbrauchsverhalten		x		
Änderbarkeit			x	
Analysierbarkeit			x	
Modifizierbarkeit			x	
Stabilität			x	
Prüfbarkeit			x	
Übertragbarkeit			x	
Anpassbarkeit			x	
Installierbarkeit			x	
Konformität			x	
Austauschbarkeit			x	

Tabelle 1: Qualitätsanforderungen

8 User Stories für Zielgruppen

8.1 Machine Learning Engineers / Data Scientists

User Story

Als Machine Learning Engineer möchte ich mit HuggingEarth vortrainierte Erdbeobachtungs-Modelle (EO-Modelle) einfach in meine Python- oder R-Workflows integrieren können, um wiederkehrende Aufgaben in meinen Projekten effizient zu automatisieren, ohne jedes Mal neue Modelle entwickeln zu müssen.

Akzeptanzkriterien:

- Ich kann nach EO-Modellen suchen, die bereits auf ähnlichen Daten (z.B. Sentinel-2) trainiert wurden.
- Die Plattform bietet mir die Möglichkeit, die Metadaten der Modelle in einem STAC-konformen Format herunterzuladen.
- Ich kann mit PySTAC die Modelle und zugehörige Metadaten einfach in mein Python-Programm integrieren und in meiner Pipeline verwenden.
- Es gibt eine Dokumentation und Beispielcode, die mir helfen, den STAC-Client zu nutzen.

8.2 Environmental Researchers

User Story

Als Umweltforscher möchte ich auf maschinelle Lernmodelle für die Fernerkundung zugreifen können, die mir bei der Klassifikation von Landnutzungsflächen und der Analyse von Zeitreihen (z.B. Veränderungen in Vegetationsflächen) helfen, um fundierte Entscheidungen für ökologische Studien und den Umweltschutz zu treffen.

Akzeptanzkriterien:

- Ich kann Modelle nach spezifischen Fernerkundungsaufgaben wie Landnutzungsklassifikation und Zeitreihenanalyse durchsuchen.
- Ich kann Modelle basierend auf ihren geographischen Einsatzbereichen und Datentypen (z.B. Sentinel-2 oder Landsat) filtern.
- Die Metadaten der Modelle enthalten detaillierte Informationen zu Anwendungsfällen, Trainingsdaten und Modellarchitektur.
- Es gibt visuelle Vorschauen, die zeigen, welche Gebiete und Daten das Modell abdeckt, um die Eignung für meine Studie besser zu beurteilen.

8.3 Geospatial Professionals

User Story

Als Geoprofessional benötige ich einsatzbereite maschinelle Lernmodelle, um georäumliche Datenverarbeitung effizient zu automatisieren, sodass ich wiederholbare Workflows schnell umsetzen und die Verarbeitung großer Datenmengen für Aufgaben wie die Klassifikation und Detektion von Mustern beschleunigen kann.

Akzeptanzkriterien:

- Ich kann aus einer Liste vortrainierter Modelle auswählen, die speziell für georäumliche Datenverarbeitungsaufgaben (z.B. Klassifikation, Anomaliedetektion) optimiert sind.
- Die Plattform bietet fertige, vortrainierte Modelle mit klaren Anweisungen zur Verwendung in der Modell-Pipeline, ohne dass zusätzliche Anpassungen erforderlich sind.
- Ich kann die Modelle und ihre Metadaten in einem standardisierten Format herunterladen und in meine bestehenden GIS-Workflows integrieren.
- Die Integration mit Python-basierten STAC-Clients ermöglicht mir die Automatisierung der Datenverarbeitung in großen Projekten, ohne die Notwendigkeit, jedes Modell manuell zu prüfen.

9 Projektzeitplan

Phase 1: Design und Architekturplanung

- **Dauer**: 2 Wochen (30. Oktober 2024 – 13. November 2024)
- **Aufgaben**:
  - Entwicklung des technischen Designs für die Plattform.
  - Festlegung der Systemarchitektur (Front-End, Back-End, Datenbankstruktur).
  - Planung der API-Struktur und STAC-Client-Integration.
  - Definition der Benutzeroberfläche und User Experience Design.
- **Meilenstein**: Abschluss der Systemarchitektur und des Designs bis **13. November 2024**.

Phase 2: Implementierung der Front-End-Komponenten

- **Dauer**: 7 Wochen (01. November 2024 – 17. Dezember 2024)
- **Aufgaben**:
  - Implementierung des Front-Ends.

  - Entwicklung der Benutzeroberfläche mit Such- und Filterfunktionen.
  - Integration von Upload- und Download-Mechanismen für Metadaten und Modelle.
  - Optimierung der Benutzerfreundlichkeit und Zugänglichkeit (Barrierefreiheit).
- **Meilenstein**: Fertigstellung des Front-Ends bis **17. Dezember 2024**.

Phase 3: Implementierung der Kernfunktionen (Back-End)

- **Dauer**: 6 Wochen (27. November 2024 – 08. Januar 2025)
- **Aufgaben**:
  - Implementierung des Back-Ends (Node.js) und der Datenbank (MongoDB).
  - Erstellung und Implementierung der RESTful API.
  - Integration der STAC-MLM-Spezifikationen in das Datenmanagement.
  - Testen der API-Schnittstellen.
- **Meilenstein**: Fertigstellung des Back-End-Systems mit funktionierenden Schnittstellen bis **08. Januar 2025**.

Phase 4: Testphase (Funktionale und nicht-funktionale Tests)

- **Dauer**: 2 Wochen (08. Januar 2025 – 22. Januar 2025)
- **Aufgaben**:
  - Durchführung von Funktionstests zur Überprüfung der Kernfunktionalitäten.
  - Durchführung von Leistungstests (Reaktionszeit, Parallelität, Datenabfragen).
  - Durchführung von Usability-Tests.
  - Behebung von Fehlern und Optimierung der Leistung.
- **Meilenstein**: Abschluss der Testphase und Behebung aller kritischen Fehler bis **22. Januar 2025**.

Phase 5: Deployment und Dokumentation

- **Dauer**: 2 Woche (15. Januar 2025 – 28. Januar 2025)
- **Aufgaben**:
  - Erstellung von Docker-Containern für Front-End und Back-End.
  - Test des Deployments auf Cloud-Plattformen (z.B. AWS, Google Cloud).
  - Erstellung der Projektdokumentation (Installationsanleitung, Benutzerhandbuch, Wartungsdokumentation).
- **Meilenstein**: Bereitstellung der Docker-Images und finale Abgabe des Projekts bis **29. Januar 2025**.

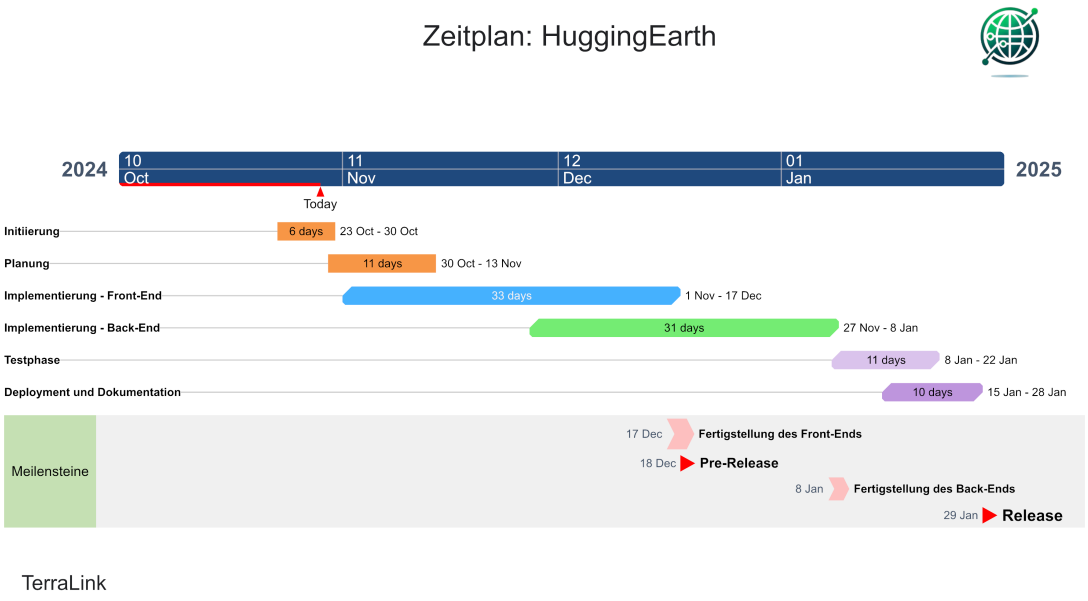

Abbildung 1: Gantt-Chart

10 Technische Abhängigkeiten

10.1 Bibliotheken und Frameworks

Für die Entwicklung des Systems werden spezifische Bibliotheken und Frameworks verwendet, die eine reibungslose und effiziente Funktionalität gewährleisten:

- **PySTAC und RSTAC**: Diese Bibliotheken werden für die Integration der STAC-Modelle und die Abfrage von Metadaten eingesetzt.
- **D3.js und WebGL**: Diese Frameworks sind für die Implementierung von Visualisierungen verantwortlich und ermöglichen hardware-beschleunigte Darstellungen.
- **Bootstrap**: Zur Gewährleistung einer responsiven Benutzeroberfläche wird das Bootstrap-Framework verwendet.

10.2 Externe APIs und Dienste

Bestimmte externe Dienste und APIs sind für die Funktion des Systems erforderlich:

- **STAC-API**: Zur Bereitstellung und Verwaltung der STAC-konformen Daten und Metadaten.
- **Mapbox oder Google Maps API**: Für die Kartendarstellung und die Visualisierung geografischer Informationen.
- **Sentinel-Daten**: Diese Daten werden als Datengrundlage für die Modellentwicklung und -validierung genutzt.

10.3 Infrastrukturanforderungen

Für die effiziente Bereitstellung und Wartung des Systems werden die folgenden Infrastrukturkomponenten benötigt:

- **Datenbank**: Eine MongoDB-Instanz zur Speicherung und Verwaltung der Metadaten und Benutzerdaten.
- **Server-Hosting**: Für den Produktiveinsatz wird ein Hosting-Dienst mit ausreichender Rechenkapazität und Skalierbarkeit empfohlen (z.B. AWS oder Google Cloud).