Implementierung (Konzept)

User Story / User Interaction (Stakeholder, Requirements, Functionalities)

Stakeholders

- **Touristen**
  Personen, die das historische Zentrum der Altstadt besuchen und Informationen über Sehenswürdigkeiten, historische Stätten und lokale Attraktionen suchen.

- **lokale Behörden**
  Kommunale Einrichtungen, die für die Pflege und Förderung des historischen Zentrums zuständig sind und genaue und zugängliche Informationen für Besucher sicherstellen.

- **Entwickler**
  Technologieexperten, die die Anwendung erstellen und warten, um sicherzustellen, dass sie den Anforderungen der Benutzer und lokalen Behörden entspricht.

- **Unternehmen**: Lokale Geschäfts- und Restaurantbesitzer, die von einem erhöhten Fußgängerverkehr und einer stärkeren Einbindung der Besucher im historischen Zentrum profitieren. Firmen, die ihre Daten in Form von Linked Open Data (LOD) zugänglich machen wollen.

- **Anwohner**: Personen, die im historischen Zentrum leben und die Anwendung möglicherweise nutzen, um mehr über ihre eigene Nachbarschaft zu erfahren oder Touristen zu helfen.

Requirements der Stakeholdern

- **Touristen**: Benötigen eine benutzerfreundliche Oberfläche, um detaillierte und genaue Informationen über historische Sehenswürdigkeiten abzurufen.

- **lokale Behörden**: Erfordern ein System, das eine genaue Datenabfrage, einfache Aktualisierung von Informationen und Analysen zu den Interaktionen der Touristen ermöglicht.

- **Entwickler**: Benötigen klare Dokumentation der Funktionalitäten, nahtlose Integration mit verknüpften offenen Daten und eine skalierbare Architektur.

- ✦ **Geschäftsinhaber**: Wünschen sich eine Plattform, um ihre Unternehmen hervorzuheben und mehr Kunden durch eine erhöhte Sichtbarkeit anzulocken.

- ✦ **Anwohner**: Streben eine nicht intrusive Integration der Anwendung in ihren Alltag an, mit der Möglichkeit, Informationen beizutragen oder zu korrigieren.

Requirements der App:

- Bereitstellung von detaillierten Informationen zu Orten basierend auf Adresse oder Koordinaten.
- Unterstützung von SPARQL-Abfragen für RDF-Daten.
- Einfache Benutzerregistrierung und -verwaltung.
- Robuste Fehlerbehandlung und klare Antworten.

Functionalities

1. Benutzerinteraktion:

- **Erkennung und Eingabe**: Die Anwendung erkennt und liest Straßenschilder oder Sehenswürdigkeiten (z.B. "9, Währingerstraße") über Texteingabe oder Geolokation (Breitengrad/Längengrad).
- **Standortbasierte Informationen**: Benutzer können entweder den Namen eines Schildes oder die Geolokation eingeben, um kontextbezogene Informationen über ihre Umgebung zu erhalten.
- **Zugriff auf historische Daten**: Bietet historische und kulturelle Daten, die aus verknüpften offenen Datenquellen abgerufen werden.
- **Benutzerfeedback**: Ermöglicht Benutzern, Feedback zu geben oder zusätzliche Informationen über Sehenswürdigkeiten beizutragen. SPARQL-Abfragen an LOD-Datenbanken senden und Ergebnisse abrufen.

2. Fähigkeiten:

- **Standort (Text/Längengrad/Latitüde) zu Verknüpften Daten**: Wandelt textuelle oder geografische Eingaben in verknüpfte Datenabfragen um, um relevante Informationen abzurufen.
- **Zugriff auf Verknüpfte Offene Daten**: Intelligentes Abfragesystem zum Zugriff auf verknüpfte offene Daten, um aktuelle und umfassende Informationen zu gewährleisten.

- **Serialisierung zu RDF**: Wandelt die abgerufenen Daten in RDF um, um strukturierte Informationsdarstellung zu ermöglichen.

3. Datenmanagement:

RDF-Graph: Speichert und verwaltet die vernetzten Daten im RDF-Format, was effiziente Abfragen und Abrufe ermöglicht.

Dynamische Aktualisierungen: Das System kann regelmäßig neue Informationen aus verschiedenen verknüpften Datenquellen aktualisieren und integrieren.

User Story
Titel: Zugriff auf historische Informationen in der Altstadt

Als Tourist möchte ich mein Mobilgerät verwenden, um ein Straßenschild abzuscannen oder den Namen eines Straßenschildes oder meinen aktuellen Standort einzugeben, **damit ich** detaillierte historische und kulturelle Informationen über die Sehenswürdigkeiten in meiner Umgebung erhalten kann.

Szenario: Abrufen von Informationen über eine Sehenswürdigkeit

1. Der **Tourist** hält sein Mobilgerät an ein Straßenschild mit der Aufschrift "9, Währingerstraße".
2. Die **Anwendung** erkennt den Text und ruft die geografischen Koordinaten ab.
3. Das **System** sendet diese Eingabe an die Fähigkeit "Standort (Text/Längengrad/Latitüde) zu Verknüpften Daten".
4. Das **intelligente Abfragesystem (Zugriff auf Verknüpfte Offene Daten)** greift auf verschiedene verknüpfte offene Datenbanken zu, um relevante Informationen zu sammeln.
5. Die abgerufenen Daten werden **in RDF serialisiert** und in einem RDF-Graphen für eine strukturierte Präsentation organisiert.
6. Der **Tourist** erhält einen umfassenden Überblick über die historische Bedeutung, bemerkenswerte Ereignisse und kulturelle Fakten zu "9, Währingerstraße" auf seinem Gerät.
7. Der **Tourist** kann Feedback geben oder zusätzliche Informationen über die Sehenswürdigkeit bereitstellen, die von den **lokalen Behörden** überprüft und in das Datensystem integriert werden können.

Service Specification (Operations, Input, Output)

1. Operation: RetrieveLODInformation (POST)

- **Description**:
  - Retrieve detailed information about a location, including historical data, nearby attractions, and other points of interest.
- **Input**:
  - `*inputName*`:
    - String (Name of the location or geographical coordinates as text, e.g., "9, Währingerstraße" or "48.213, 16.355") ○ `*query*`:
    - String (SPARQL query to retrieve linked data, automatically generated from the input) ○ `*endpoint*` (optional):
    - String (URL of the LOD database endpoint) ● **Output**:
  - *LODData`:*
    - Object (Details about the location, including history, nearby cafes, restaurants, bars, and attractions)

2. Operation: ConvertToLinkedDataQuery

- **Description**:
  - Converts the detected location data into a SPARQL query for retrieving linked data.
- **Input**:
  - `*locationData*`:
    - Object (Processed input data including text and geolocation information) ● **Output**:
  - *`linkedDataQuery`:*
    - String (Formatted SPARQL query to retrieve linked data)

3. Operation: AccessLinkedOpenData

- **Description**:
  - Executes the SPARQL query to retrieve information from linked open data sources.
- **Input**:
  - *`linkedDataQuery`:*
    - String (SPARQL query) ● **Output**:
  - *`rawData`:*
    - Object (Raw data retrieved from linked open data sources)

4. Operation: SerializeToRDF

- **Description**:

    - Converts the retrieved raw data into RDF format.
- **Input**:
    - *`rawData`:*
        - Object (Raw data from LOD sources) ●

Output:

- *`rdfGraph`:*
    - Object (Structured RDF graph)

5. Operation: ProvideUserInteraction

- **Description**:
    - Manages user interaction by displaying the retrieved information and collecting feedback.
- **Input**:
    - *`rdfGraph`:*
        - Object (Structured RDF graph) ○

    `userFeedback` (optional):
        - String (User feedback) ●

Output:

- *`displayData`:*
    - Object (Formatted data for display) ○

    `feedbackAcknowledgment`:
    - String (Acknowledgment of received feedback)

Channels

- **Topic**: `citizen.development.1.0.action.{inputName}.retrieve.lod.information`
    - **Operation**: `retrieveLODInformation`
    - **Description**: Retrieves level of detail information about a specific location.
    - **Input**: `inputName`, `query`
    - **Output**: `LODData`

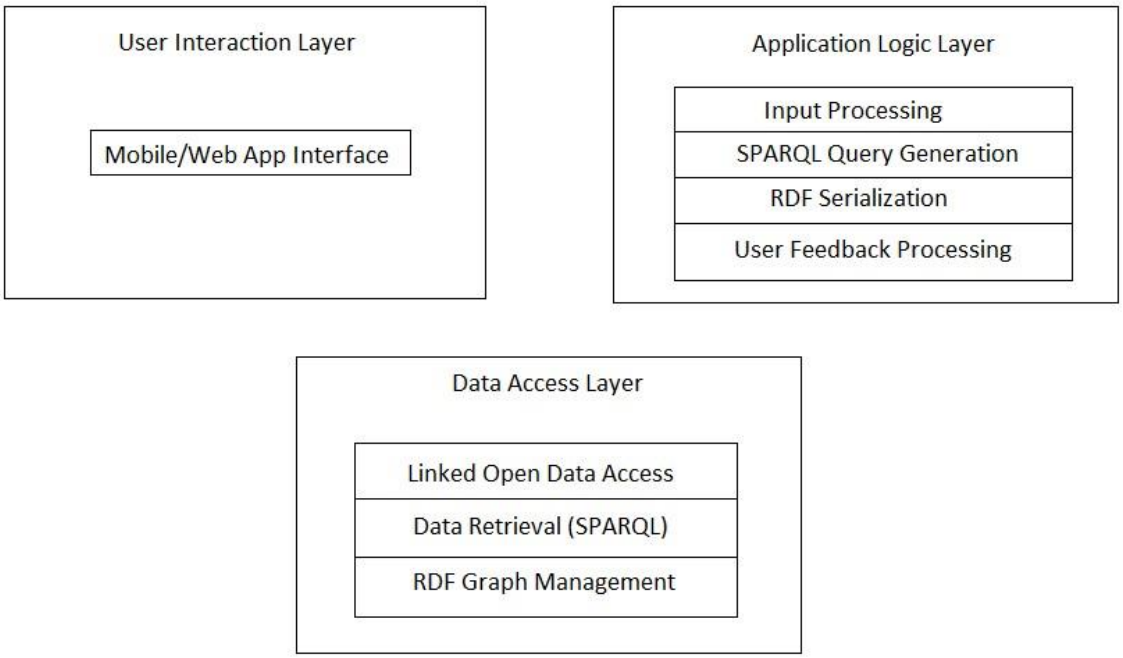
- **Topic**: `citizen.development.1.0.event.{endpoint}.{query}.access.lod`
    - **Operation**: `accessLODData`
    - **Description**: Accesses LOD databases and performs intelligent queries.
    - **Input**: `query`, `endpoint`
    - **Output**: `rdfBasedResults`

Technology Evaluation (Architecture, Functional Capabilities, Deployment with Dependencies, Legal Considerations)

1. Architekturübersicht

Die Systemarchitektur für die Citizen Development API umfasst mehrere Schlüsselkomponenten und Technologien, um die Abfrage und Verarbeitung von LOD zu ermöglichen. Die Architektur kann in folgende Schichten unterteilt werden:

User Interaction Layer: Verwaltet die Benutzeroberfläche und Interaktionen.
Application Logic Layer: Beinhaltet die Kernlogik zur Verarbeitung von Eingaben, Abfragen von LOD-Quellen und Formatierung der Ausgaben.
Data Access Layer: Handhabt den Zugriff auf LOD-Datenbanken und verwaltet die Datenserialisierung und -deserialisierung.



Detaillierter Datenfluss

- **Benutzereingabe:**
  - Der Benutzer gibt Daten über eine Web-/Mobile-App ein.
  - Die Daten werden über einen API-Aufruf an das Backend gesendet.
- **Verarbeitung:**
  - Die API Gateway empfängt die Eingabe.
  - Der Eingabeprozessor validiert und analysiert die Eingabe.
  - Der SPARQL-Abfragedienst generiert und führt eine SPARQL-Abfrage aus.
- **Datenabruf:**
  - Der SPARQL Query Service greift über einen SPARQL Endpoint auf die LOD-Datenbank zu.

    - RDF-Daten werden abgerufen und an den RDF-Serialisierungs-Service gesendet.
- **Serialisierung:**
    - RDF-Daten werden in ein geeignetes Format konvertiert (JSON-LD).
    - Die serialisierten Daten werden zurück an das API Gateway gesendet.
- **Antwort:**
    - Das API Gateway gibt die formatierten Daten an die Benutzerinteraktionsebene zurück.
    - Die Web-/Mobile-App zeigt die Informationen dem Benutzer an.

2. Funktionale Fähigkeiten und Technologieentscheidungen

1. User Interaction Layer

Funktionale Fähigkeiten:

- Verarbeitung von Benutzereingaben (Text oder Koordinaten)
- Anzeige der abgerufenen Informationen
- Sammlung und Verarbeitung von Benutzerfeedback

Technologieentscheidungen:

- Frontend-Framework: Angular 3 für den Aufbau interaktiver Weboberflächen. 2. Application Logic Layer

Funktionale Fähigkeiten:

- Verarbeitung von Benutzereingaben zur Generierung von SPARQL Queries
- Verwaltung der Ausführung von SPARQL Queries
- Serialisierung von Daten in RDF-Format
- Integration von Benutzerfeedback in das System

Technologieentscheidungen:

- Programmiersprache: Java für Backend-Sevices aufgrund ihrer umfangreichen Ökosysteme und Unterstützung für asynchrone Operationen.
- SPARQL-Bibliotheken: `Apache Jena` für Java zur Verarbeitung von SPARQL-Abfragen.
- RDF-Bibliotheken: `RDF4J` für Java zur Verwaltung von RDF-Daten. 3. Data Access Layer

Funktionale Fähigkeiten:

- ○ Zugriff auf LOD-Datenbanken
- ○ Ausführung intelligenter Abfragen zur Abholung verknüpfter Daten
- ○ Verwaltung von RDF-Grafiken und Datensätzen

Technologieentscheidungen:

- ○ Datenbankzugriff: Apache Jena Fuseki zur Verwaltung von SPARQL Endpoints.
- ○ RDF-Datenverwaltung: Blazegraph/GraphDB für skalierbare RDFDatenspeicherung und Abfrageverarbeitung.
- ○ Nachrichtenbroker: Apache Kafka zur Handhabung von EchtzeitDatenströmen und zur Gewährleistung einer robusten Kommunikation zwischen Diensten.

Schlüssel-Funktionsfähigkeiten

- **Eingabeverarbeitung**: Unterstützt Text (Adresse) oder Koordinaten als Eingabe.
- **Datenabruf**: Effiziente Abfrage von LOD-Quellen mit SPARQL.
- **Datenserialization**: Konvertiert RDF-Daten in benutzerfreundliche Formate. ● **Benutzerinteraktion**: Interaktive Oberflächen für Web- und Mobilnutzer.
- **Echtzeitdatenverarbeitung**: Nutzt Apache Kafka zur Verarbeitung von Echtzeitdatenströmen und zur Gewährleistung responsiver Interaktionen.

4. Developing mit Abhängigkeiten

Deployment Dependencies

- **Frontend:**
  - ○ Thymeleaf Template ●

Backend:

- ○ Java: Benötigt Laufzeitumgebungen für Java.
- ○ SPARQL Query Services: Benötigt Bibliotheken (RDF4J/Apache Jena)
- ○ RDF Databases: Benötigt Installation und Konfiguration von (Apache Jena Fuseki/Blazegraph/GraphDB)

Deployed wird der Microservice über Docker.

Continuous Integration und Continuous Deployment (CI/CD)

- CI/CD-Tools: GitLab CI zur Automatisierung von Build-, Test- und Bereitstellungsprozessen.
- Container-Registry: Docker Hub (Portainer) zur Speicherung von ContainerImages.

4. Rechtliche Überlegungen

1. Datenschutz und -sicherheit

- Compliance:
  - Einhaltung von GDPR, CCPA und anderen Datenschutzvorschriften sicherstellen.
  - Implementierung von Datenanonymisierung und -verschlüsselung für sensible Informationen.
- Nutzerzustimmung:
  - Einholung der Nutzerzustimmung für die Erfassung von Standortdaten und persönlichen Informationen.
  - Bereitstellung klarer Datenschutzrichtlinien und Nutzungsbedingungen.

2. Lizenzierung

- Open-Source-Bibliotheken:
  - Überprüfung der Lizenzen der verwendeten Open-Source-Bibliotheken und -Frameworks (MIT/Apache 2.0/ GPL).
  - Sicherstellung der Einhaltung der Lizenzanforderungen und beschränkungen.

3. Nutzung von Linked Open Data (LOD)

- Datenlizenzierung:
  - Überprüfung der Lizenzbedingungen der abgefragten LOD-Quellen.
  - Sicherstellung, dass die Nutzung den Bedingungen entspricht, insbesondere für kommerzielle Zwecke.

4. Geistiges Eigentum

- Eigentum:
  - Sicherstellung, dass der gesamte entwickelte Code und das geistige Eigentum ordnungsgemäß dokumentiert und geschützt sind.
  - Verwendung geeigneter Lizenzen für proprietäre Softwarekomponenten.

5. Sicherheit

- Authentifizierung und Autorisierung:
  - Implementierung sicherer Authentifizierung (OAuth 2.0/JWT) für den API-Zugriff.
  - Sicherstellung einer ordnungsgemäßen rollenbasierten

Zugriffskontrolle (RBAC) zur Beschränkung des Zugriffs auf sensible Operationen.

- Datensicherheit:
  - Verwendung von HTTPS für alle Kommunikationswege zwischen Client, Server und Datenbanken.
  - Regelmäßige Aktualisierung von Abhängigkeiten zur Minderung von Sicherheitslücken.
  - Durchführung regelmäßiger Sicherheitsüberprüfungen und Penetrationstests.

Erweiterungen und Verbesserungen

Skalierbarkeit:

- Einsatz von Load-Balancern und Skalierungsmechanismen, um eine hohe Verfügbarkeit und Leistung der API sicherzustellen.
- Nutzung von Caching-Strategien, um häufige Anfragen effizienter zu bedienen.

Sicherheitsverbesserungen:

- Implementierung von Rate-Limiting, um Missbrauch der API zu verhindern.
- Regelmäßige Sicherheitsüberprüfungen und Penetrationstests zur Identifizierung und Behebung von Sicherheitslücken.

Erweiterte Funktionalitäten:

- Hinzufügen von Filtermöglichkeiten bei Abfragen, z.B. nach Preiskategorie, Öffnungszeiten, Bewertungen.
- Implementierung von Benutzerbewertungen und Kommentaren für Orte, um die Interaktivität und den Nutzen der API zu erhöhen.
- Unterstützung weiterer Datenformate und Schnittstellen, z.B. OData.

Dokumentation und Support:

- Bereitstellung einer umfassenden Dokumentation der API, einschließlich Beispielanfragen und -antworten.