**UrHomie**

A diagram of a diagram

AI-generated content may be incorrect.

***Funționalitate Autentificare/Autorizare***

*A diagram of a network

AI-generated content may be incorrect.*

*UserAuthMicroservice:*

1. Introducere

Acest proiect implementează un microserviciu dedicat autentificării utilizatorilor, utilizând tehnologii moderne precum gRPC pentru comunicare inter-servicii, JSON Web Tokens (JWT) pentru managementul sesiunilor și biblioteca Peewee pentru interacțiunea cu baza de date MySQL. Scopul principal al acestui microserviciu este de a valida credențialele utilizatorilor, de a genera token-uri de acces și de a verifica validitatea acestora.

1. Arhitectura Generală a Sistemului

Proiectul este structurat în mai multe straturi:

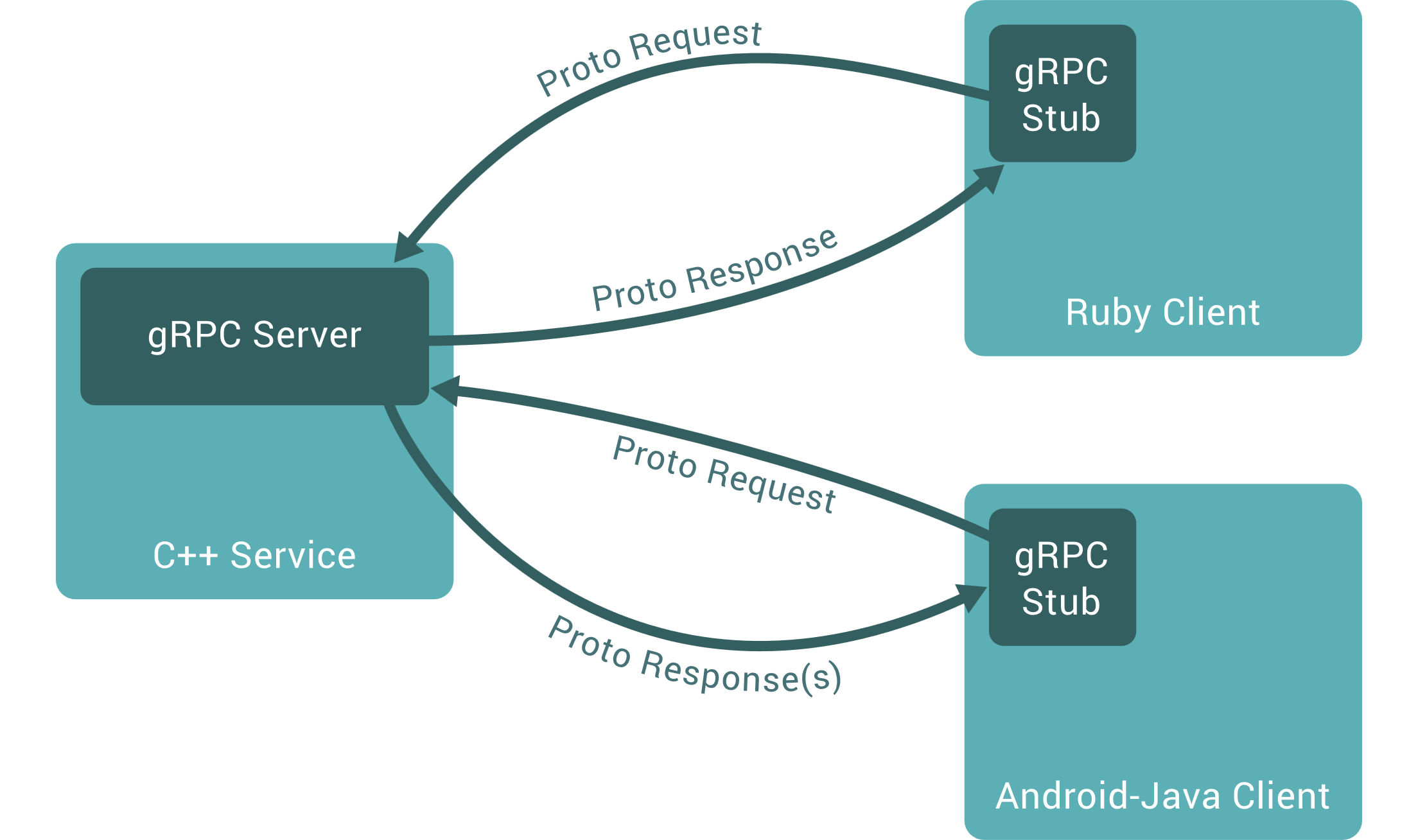
* **Stratul de Prezentare**: Expune serviciile printr-un server gRPC, permițând interacțiunea cu clienții.
* **Stratul de Business**: Conține logica de autentificare și gestionare a token-urilor JWT, cât și generarea interfețelor gRPC server/clientț
* **Stratul de Persistență**: Gestionează operațiunile asupra bazei de date, fiind realizat prin ORM-ul Peewee, care interacționează cu o bază de date MySQL.
* **Containerizare**: Aplicația este containerizată folosind Docker, facilitând astfel instalarea și rularea consistentă pe diferite medii.

A screenshot of a computer

AI-generated content may be incorrect.Această arhitectură modulară permite o dezvoltare scalabilă și ușor de întreținut, separând clar responsabilitățile fiecărui modul.

1. Componentele Microserviciului
   1. Comunicarea prin gRPC

Frameworkul gRPC reprezintă o tehnologie modernă pentru dezvoltarea de aplicații distribuite, fiind un sistem de apeluri de proceduri la distanță (RPC) dezvoltat de Google. Bazat pe Protocol Buffers pentru serializarea datelor și pe HTTP/2 pentru transport, gRPC permite definirea clară a interfețelor prin fișiere .proto, asigurând o comunicare strict tipizată între client și server. Această abordare nu doar că simplifică generarea automată a codului pentru ambele părți, dar oferă și beneficii în ceea ce privește performanța datorită suportului pentru multiplexare, streaming și compresie nativă, toate acestea contribuind la reducerea latenței și la optimizarea resurselor în sistemele distribuite.

În contextul proiectului meu, gRPC se dovedește util prin capacitatea sa de a facilita comunicarea rapidă și sigură între microservicii, esențială pentru un sistem de autentificare robust. Utilizarea gRPC permite implementarea eficientă a metodelor de autentificare printr-un contract clar definit, reducând ambiguitățile și erorile de integrare între diferitele componente ale aplicației. Mai mult, suportul pentru diverse limbaje de programare și integrarea facilă cu alte tehnologii moderne contribuie la scalabilitatea și menținerea pe termen lung a sistemului.

* **Fișierul *generate\_from\_proto.bash***  
  Acest script automatizează generarea interfetelor client/server din fișierele .proto. Utilizând modulul grpc\_tools.protoc din Python, scriptul specifică directorul sursă și directorul de ieșire pentru fișierele generate.



* **Fișierul *user\_auth.proto***  
  Definirea interfeței de autentificare este realizată prin specificații în format Protocol Buffers (proto3). Acest fișier definește:
  + Serviciul UserAuthentication cu două proceduri:
    - LogIn – primește un mesaj LogInRequest și returnează LogInResponse (token-ul JWT).
    - ValidateJwt – primește un mesaj ValidateJwtRequest și returnează ValidateJwtResponse (starea token-ului).
  + Mesajele folosite pentru cereri și răspunsuri, cu câmpuri relevante precum email, parolă, token JWT, roluri și eventuale mesaje de eroare.
  1. Serviciul JWT

JWT (JSON Web Token) reprezintă un standard deschis pentru transmiterea în siguranță a informațiilor între părți, folosind un obiect JSON compact și URL-safe. Un JWT este compus din trei componente principale: header, payload și signature. Header-ul specifică tipul token-ului și algoritmul de semnare (de exemplu, HS256), payload-ul conține declarații (claims) precum identitatea utilizatorului, rolurile și data expirării, iar signature-ul este generat prin aplicarea algoritmului specific pe datele combinate din header și payload, folosind o cheie secretă. În esență, semnarea JWT conform specificațiilor JWS (JSON Web Signature) asigură integritatea și autenticitatea token-ului, garantând că datele nu au fost modificate în timpul transmisiei.

În cadrul proiectului, componenta JWT este esențială pentru gestionarea autentificării și autorizării utilizatorilor, folosind biblioteca PyJWT pentru operațiuni criptografice. La autentificare, sistemul generează un token JWT care conține informații despre utilizator, precum email-ul și rolurile acestuia, stabilind totodată un timp de expirare și un identificator unic (JTI). Acest token este transmis clientului și utilizat ulterior pentru accesarea resurselor protejate, fiind inclus în fiecare solicitare către server. La primirea unei cereri, serverul validează token-ul pentru a verifica autenticitatea și drepturile utilizatorului, gestionând eventuale excepții legate de expirarea sau invaliditatea acestuia.

* 1. Configurare si Conectivitate la Baza de Date

MySQL este un sistem de gestionare a bazelor de date relaționale (RDBMS) open-source, cunoscut pentru fiabilitatea, performanța și scalabilitatea sa. În contextul proiectului tău, utilizarea MySQL este o alegere potrivită datorită capacității sale de a gestiona eficient datele utilizatorilor, inclusiv credențiale, roluri și permisiuni. Structura relațională oferă integritate și consistență datelor, iar suportul pentru tranzacții asigură securitatea operațiunilor critice, precum autentificarea și gestionarea accesului. De asemenea, MySQL se integrează bine cu Peewee, ORM-ul folosit în proiect, facilitând interacțiunea cu baza de date printr-o interfață simplificată. Fiind un sistem matur și optimizat pentru performanță, MySQL permite gestionarea unui volum mare de date și susține cerințele unui microserviciu de autentificare, garantând răspunsuri rapide și un management eficient al sesiunilor utilizatorilor.

Fișierul *db\_config.py* definește parametrii de conectare la baza de date MySQL, utilizând variabile de mediu pentru flexibilitate în diferite medii de rulare (ex.: dezvoltare, testare, producție). Conexiunea se realizează prin intermediul clasei MySQLDatabase oferită de Peewee. Această abordare permite izolarea detaliilor de configurare și o mai bună gestionare a securității.

Pentru gestionarea bazei de date, proiectul include o serie de scripturi SQL:

* **1-grant.sql**:
  + Creează baza de date user-auth dacă aceasta nu există.
  + Acordă privilegii de SELECT, INSERT, UPDATE și DELETE pentru utilizatorul user-auth-manager, asigurând un nivel adecvat de securitate.
* **2-create.sql**:
  + Definește tabelele esențiale: app\_user, role și app\_user\_role.
  + Specifică cheile primare, restricțiile de unicitate și relațiile între tabele prin intermediul cheilor externe, asigurând integritatea referențială.
* **3-insert.sql**:
  + Introduce un utilizator de tip administrator, împreună cu hash-ul parolei (utilizând bcrypt).
  + Populează tabelul role cu rolurile predefinite.
  + Asociază rolul de administrator cu utilizatorul prin tabelul app\_user\_role.

Această abordare modulară a scripturilor permite un control precis asupra etapelor de configurare a bazei de date și facilitează procesele de migrare, rollback si de initializare a bazei de date in procesul de containerizare.

Interacțiunea cu baza de date este realizată prin ORM-ul Peewee, care oferă o abordare orientată pe obiecte pentru gestionarea entităților.

* **Modelele de Date**:
  + **AppUser**: Reprezintă utilizatorii aplicației. Conține atribute precum id, email și hashed\_password.
  + **Role**: Definește rolurile disponibile în sistem (ex.: platform-manager, client, service-provider).
  + **AppUserRole**: Model de legătură între AppUser și Role, implementând relația many-to-many.
  + **BaseModel**: Clasa de bază care asociază toate modelele la conexiunea de bază de date definită în configurație.
* **Repository-ul Utilizatorilor** (app\_user\_repo.py):  
  Această componentă are rolul de a abstractiza operațiunile asupra bazei de date:
  + **validate\_credentials(email, password)**:
    - Verifică existența utilizatorului în baza de date și validează parola folosind bcrypt pentru compararea hash-urilor.
  + **get\_user\_roles(user\_id)**:
    - Efectuează interogări complexe de tip JOIN între tabelele role, app\_user\_role și app\_user pentru a obține lista de roluri asociate unui utilizator.

Aceste componente permit o separare clară a responsabilităților între logica de afaceri și accesul la date, facilitând astfel mentenanța și eventualele extinderi ale sistemului.

**!! Capitol despre RabbitMQ !!**

* 1. Prezentarea Serviciului gRPC

Serverul gRPC funcționează ca un punct central de gestionare a autentificării utilizatorilor, expunând servicii printr-o interfață bine definită în fișierul **user\_auth.proto**. Acesta implementează două metode esențiale: **LogIn**, pentru autentificarea utilizatorilor și generarea unui token JWT, și **ValidateJwt**, pentru verificarea validității token-urilor. Serverul este construit folosind **gRPC Python** și rulează pe un model asincron bazat pe **ThreadPoolExecutor**, permițând gestionarea eficientă a mai multor cereri simultane. La primirea unei solicitări, serverul procesează datele, interacționează cu baza de date MySQL prin intermediul ORM-ului Peewee și returnează un răspuns structurat conform protocolului gRPC. Această abordare asigură comunicare rapidă și sigură între microservicii, reducând latența și îmbunătățind scalabilitatea aplicației.

**!! Inserare imagine detalii (url, raspuns-uri etc)call-uri gRPC !!**

* 1. Containerizare și Implementare (Docker)

Containerizarea este o metodă de virtualizare la nivel de sistem de operare care permite rularea aplicațiilor izolate în containere, împreună cu toate dependințele necesare. **Docker** facilitează acest proces prin crearea de **imagini Docker**, care reprezintă șabloane reutilizabile ce conțin codul aplicației, bibliotecile și configurațiile necesare pentru rularea acesteia într-un mediu controlat.

Fișierul **Dockerfile** definește mediul de rulare al microserviciului descris, asigurând portabilitatea și uniformizarea implementării. Se bazează pe imaginea **Ubuntu 22.04**, peste care sunt instalate **Python 3.11** și pachetele necesare pentru dezvoltare și compilare. Se configurează un mediu virtual pentru izolarea dependențelor, iar variabilele de mediu relevante, precum **PATH** și **PYTHONPATH**, sunt setate pentru a facilita rularea aplicației. Codul sursă și fișierul **requirements.txt** sunt copiate în container, iar bibliotecile esențiale – precum **grpcio, PyJWT, peewee, mysqlclient** și **bcrypt** – sunt instalate pentru a permite comunicarea gRPC, gestionarea JWT și interacțiunea cu baza de date MySQL. În final, containerul este configurat să ruleze automat serverul gRPC, oferind un mediu standardizat și reproductibil pentru desfășurarea microserviciului.

Implementari viitoare:

* “Ascundere” SECRET\_KEY (momentan hardcoded, in jwt\_service.py);
* Rotate Refresh Token;
* “Ai uitat parola” implementare;
* ….