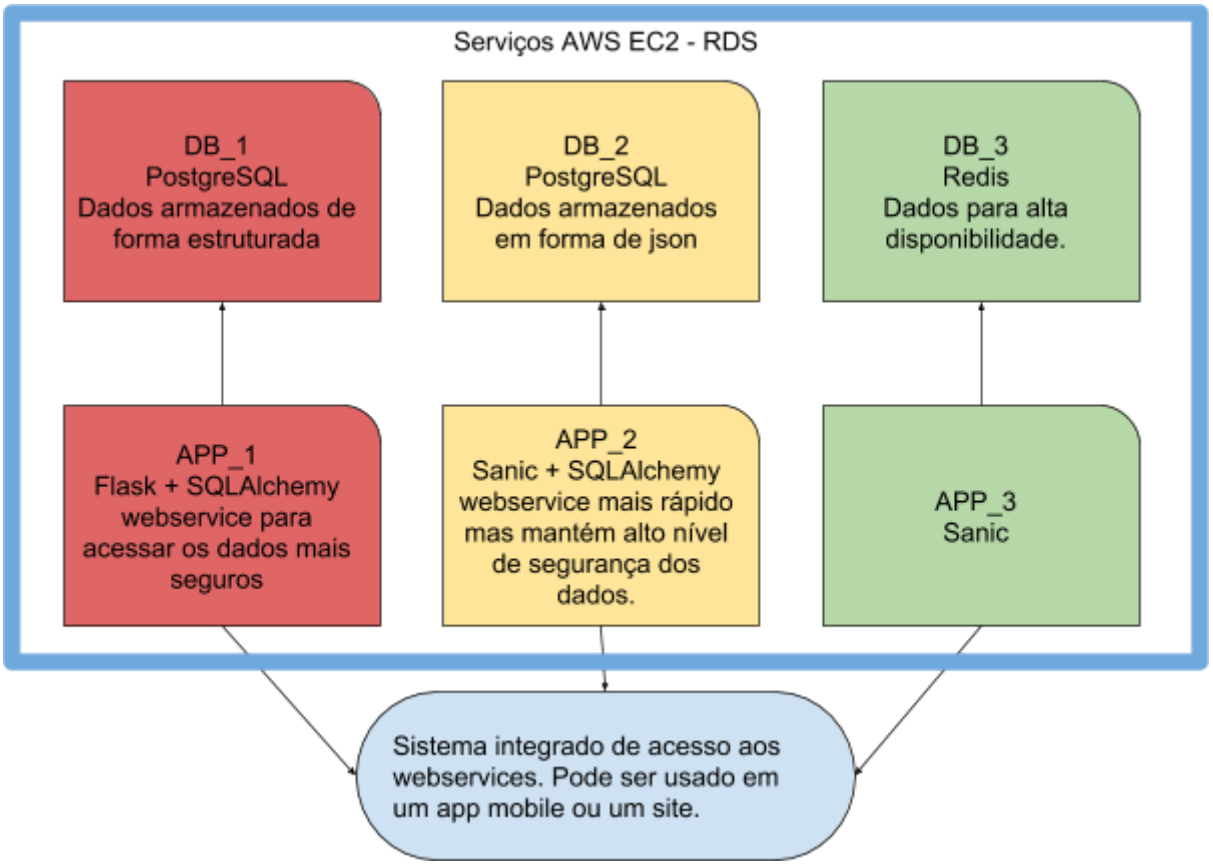
Diagrama da Arquitetura



O banco de dados/serviço 1, que possui as informações mais críticas, será armazenado em um banco de dados relacional. Isto garante a segurança nas transações. O banco de dados escolhido foi o PostgreSQL, conhecido por sua estabilidade e robustez. Sua estrutura de dados deve ser fortemente tipada e protegida. Para isso foi utilizado o framework SQLAlchemy, para garantir a manipulação das informações de forma segura. Além disso o framework fornece uma interface mais fácil e intuitiva para se trabalhar com os dados, evitando assim longas cadeias de comando SQL. Isto pode garantir um desenvolvimento bem mais rápido para sistemas que necessitam de banco de dados relacional.
Para servidor web foi utilizado o framework Flask, que pode ser utilizado tanto como servidor restfull, como servir dados diretamente. Apesar de não ser o framework web mais rápido, ele é muito seguro e rápido para o desenvolvimento de novas aplicações.

Para o banco de dados/serviço 2, que possui um nível de segurança intermediário, os dados serão armazenados também em um banco de dados relacional, mas a forma de armazenamento não será realizado em tabelas com relacionamentos complexos, que deixam as consultas mais lentas. Os dados serão armazenados em uma tabela, organizados em formato json. Isto garante um desenvolvimento mais livre, dando flexibilidade para a equipe trabalhar com dados que podem evoluir rápido, sem se preocupar com migration ou modificações na estrutura de dados. Além disso as consultas

são mais rápidas, retornando prontamente uma lista de dados com apenas uma simples consulta. Isto irá facilitar e agilizar as consultas de machine learning que precisam ser executados neste banco/serviço. O banco de dados escolhido foi o PostgreSQL. A manipulação dos dados será feita utilizando o framework SQLAlchemy. Isto garante a segurança nas transações, mantém a atomicidade, mesmo utilizando os dados fora de um contexto convencional de tabelas relacionadas. No framework web, foi utilizado Sanic, que possui um desenvolvimento muito parecido com o Flask, mas tem um desempenho muito superior. Além disso o Sanic foi sempre projetado para trabalhar com requisições assíncronas, multithead e não bloqueante. Seu desenvolvimento está evoluindo de forma rápida e está ganhando segurança e estabilidade em pouco tempo.

E por fim, no banco de dados/serviço 3 foi utilizado Redis, que utiliza armazenamento chave-valor em memória. Com foco em velocidade não será usado SQLAlchemy ou qualquer outro framework para manipulação dos dados. Além disso como o Redis não é um banco de dados relacional, muitas das vantagens em se utilizar um ORM não seriam aproveitadas. Como framework web será mantido o desenvolvimento com Sanic. Quando bem balanceado, utilizando o máximo de operações assíncronas e multithread, pode-se ter um desempenho excelente. Além disso manter toda a equipe trabalhando com a mesma linguagem e com tecnologias parecidas, aumenta a produtividade e reduz prazos e custos durante o desenvolvimento.

Todos os webservices podem ser acessados e consumidos por várias formas, desde aplicativos mobile, sites, api’s, etc.