PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

Curso: Ciência de Dados e Inteligência Artificial

Disciplina: Programação WEB

Professor: Fernando Luiz de Almeida Silveira

Grupo: AIgents – Enzo Ambrósio da Costa - 24008773
João Vitor Guedes del Ducca - 24012015
Thomaz Pinheiro Dacorso - 24012310
Victor Manhãis França - 24010801
Vitor Riki Araujo Furuta - 24008775

DOCUMENTAÇÃO 3 PARCIAL PROJETO AIgents

1. Etapas da Documentação

1.1. Levantamento de Requisitos

Objetivos:

- Criar uma plataforma web que concentre diversos agentes de inteligência artificial em um único ambiente digital.
- Permitir que usuários utilizem agentes existentes ou criem novos agentes.
- Democratizar o acesso à IA, reduzindo barreiras de usabilidade e descoberta.

Funcionalidades Principais:

- Cadastro e login de usuários.
- Busca e categorização de agentes de IA.
- Avaliação e feedback dos agentes.
- Upload de agentes.
- Visualização de dados de uso para criadores.

Restrições Técnicas:

- UI feita com HTML, CSS e JS (futuramente o uso de algum framework de reatividade).
- API feita com framework flask do Python.
- API base feita com Golang Gin.
- Banco de dados relacional (PostgreSQL) para cadastro e gerenciamento de informações.
- Hospedagem em servidor acessível via navegador.
- Compatibilidade inicial com desktop.

Personas:

Persona 1	Usuário Comum	Busca aumentar produtividade pessoal ou explorar IA para entretenimento.
Persona 2	Empresário/Startup	Utiliza agentes de IA para otimizar processos.
Persona 3	Criador de Agentes	Desenvolvedor independente interessado em disponibilizar e monetizar seus modelos.

Requisitos Funcionais:

RF01	Cadastro/login	Página para cadastro/login dos usuários.
RF02	Busca de agentes por categorias	Sistema de filtro por categorias dos agentes.
RF03	Interagir com agentes existentes	Utilização dos agentes disponíveis na página pelos usuários.
RF04	Upload de agentes por usuários autorizados	Ambiente para dar upload dos agentes por usuário com assinatura (implementação futura do sistema de pagamento).
RF05	Avaliações e comentários	Aba vinculada a cada agente na qual os usuários podem avaliar o agente para que o criador possa fazer melhorias.
RF06	Aba do criador	Aba na qual o criador de agentes pode ver seus projetos e suas avaliações.
RF07	Espaço da comunidade	Ambiente para os usuários do site interagirem entre si.

Requisitos Não Funcionais:

RNF01	Interface intuitiva e responsiva	A plataforma deve ser de fácil uso, com navegação clara e acessível.
RNF02	Segurança no armazenamento de dados e transações	O sistema deve garantir a proteção das informações dos usuários e a confiabilidade das operações financeiras, utilizando boas práticas de criptografia e políticas de privacidade.
RNF03	Suporte a múltiplos usuários simultaneamente	A plataforma deve manter seu desempenho mesmo com vários acessos e interações ao mesmo tempo.
RNF04	Escalabilidade para comportar crescimento da base de agentes	A infraestrutura deve permitir expansão conforme a demanda, garantindo que o aumento no número de agentes e usuários não comprometa a qualidade do serviço.

1.2. Planejamento e Arquitetura

Escopo:

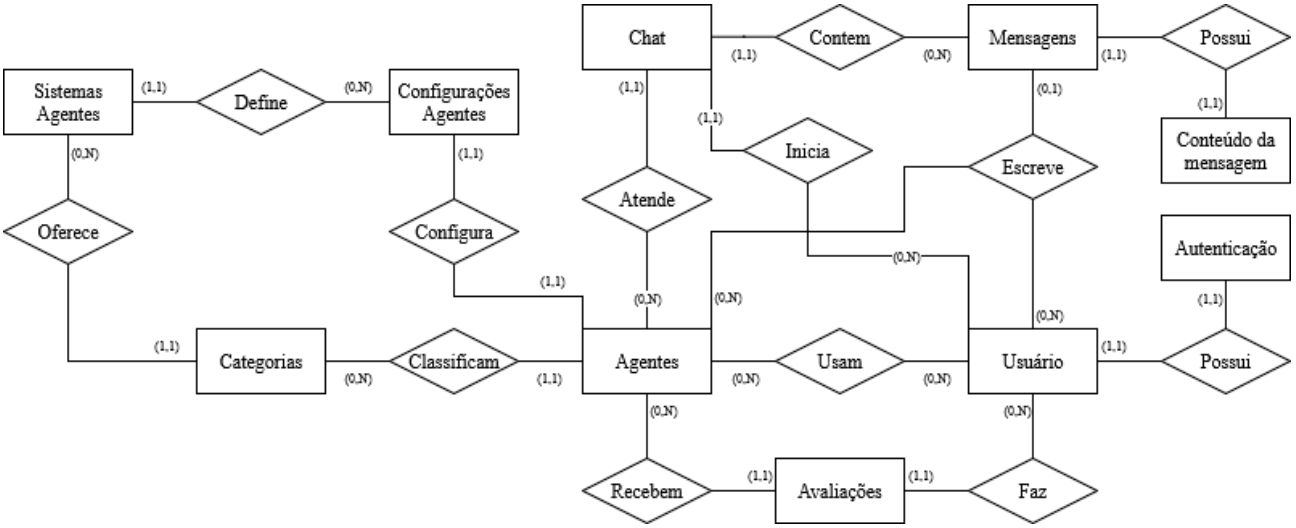
- Desenvolver versão web funcional com autenticação, navegação básica, criação e uso de agentes.
- Priorizar a acessibilidade e confiança do usuário.
- Garantir espaço para expansão de funcionalidades futuras.

Tecnologias e Ferramentas:

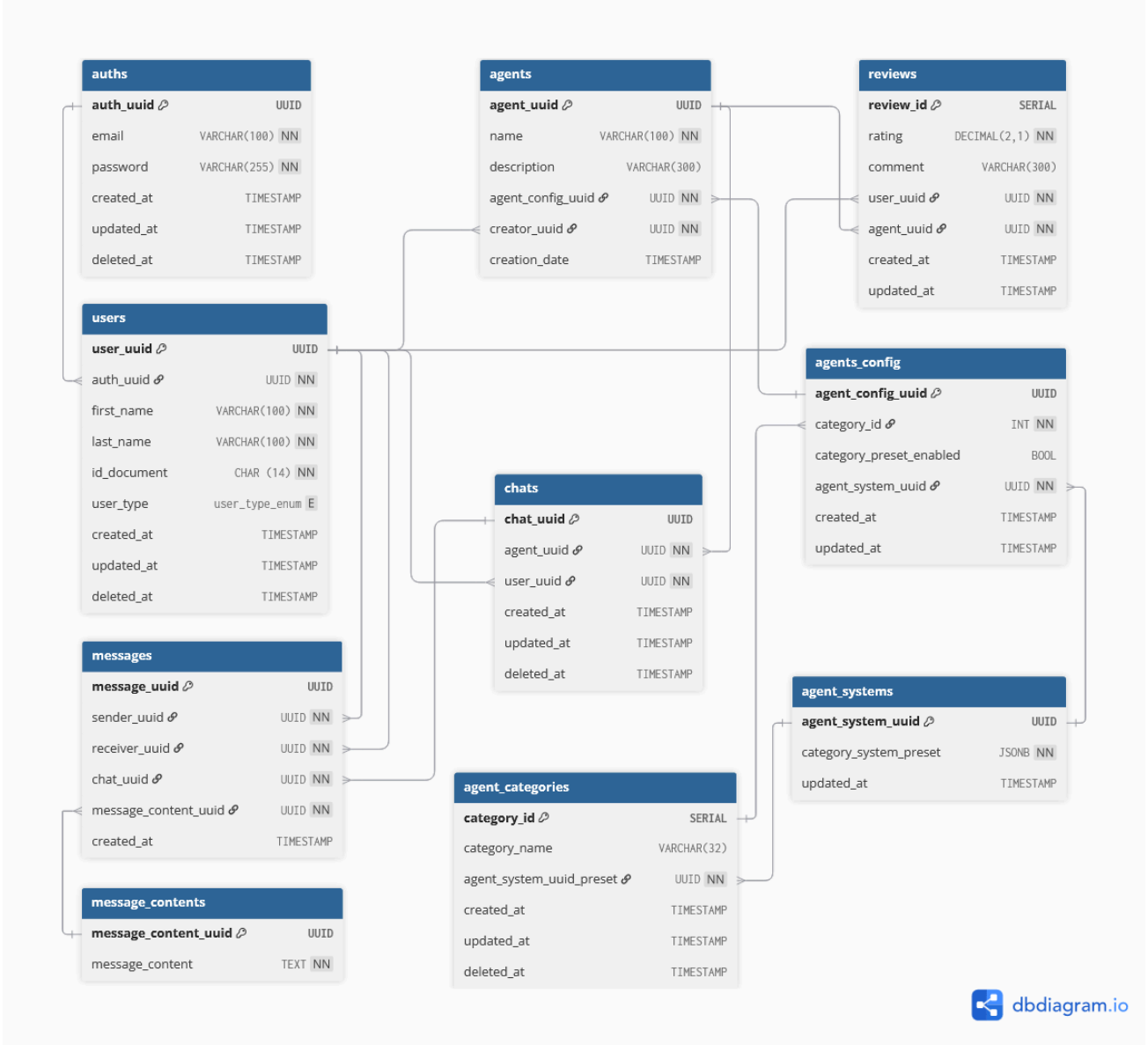
- Frontend: HTML, CSS e JS.
- Backend: Python (LangChain e Flask), Golang (Gin)
- Banco de Dados: PostgreSQL.
- Controle de Versão: GitHub.
- Design: Figma para protótipos.

Diagramas:

- Modelo conceitual do banco de dados



- Modelo lógico do banco de dados



Metodologia de desenvolvimento:

- Scrum, sprints de 2 semanas.

Sprint 1 (19/08 a 26/08)	● Ideação	Concluído
	● Prototipação	Concluído
Sprint 2 (26/08 a 09/09)	● Criação das telas inicias em HTML	Concluído
	● Modelagem do Banco de Dados	Concluído
Sprint 3 (09/09 a 23/09)	● Criação do Banco de Dados	Concluído
	● Implementação de mais telas e estilização com CSS	Concluído
Sprint 4 (23/09 a 07/10)	● Criação da API base	Concluído
	● Finalização das telas do projeto	Concluído
	● API da parte de IA	Em andamento
Sprint 5 (07/10 a 30/10)	● Implementação do JavaScript	Concluído
	● Integração frontend, backend e banco de dados	Em andamento
Sprint 6 (30/10 a 13/11)	● Testes e validações	Em andamento
Sprint 7 (13/11 a 27/11)	● Finalização do projeto	A fazer

1.3. Desenvolvimento

Estrutura de Código:

- Frontend

```
ui/
├── estilos/          # Módulo de Estilos (CSS)
│   ├── styles.css    # Arquivo principal com imports
│   ├── home.css      # Estilos da landing page
│   ├── log_sign.css  # Estilos de login/cadastro
│   ├── agents.css    # Estilos da página de agentes
│   ├── projects.css  # Estilos de projetos
│   └── about.css     # Estilos sobre nós
│
├── Index.html/       # Módulo de Estrutura (HTML)
│   ├── Home.html
│   ├── Login.html
│   ├── Sign-up.html
│   ├── Agents.html
│   ├── Myprojects.html
│   └── About.html
│
└── Scripts/          # Módulo de Lógica (JavaScript)
    ├── scriptLogin.js
    ├── scriptSign.js
    └── projects.js
```

- Backend

API Base - Golang

api/base/

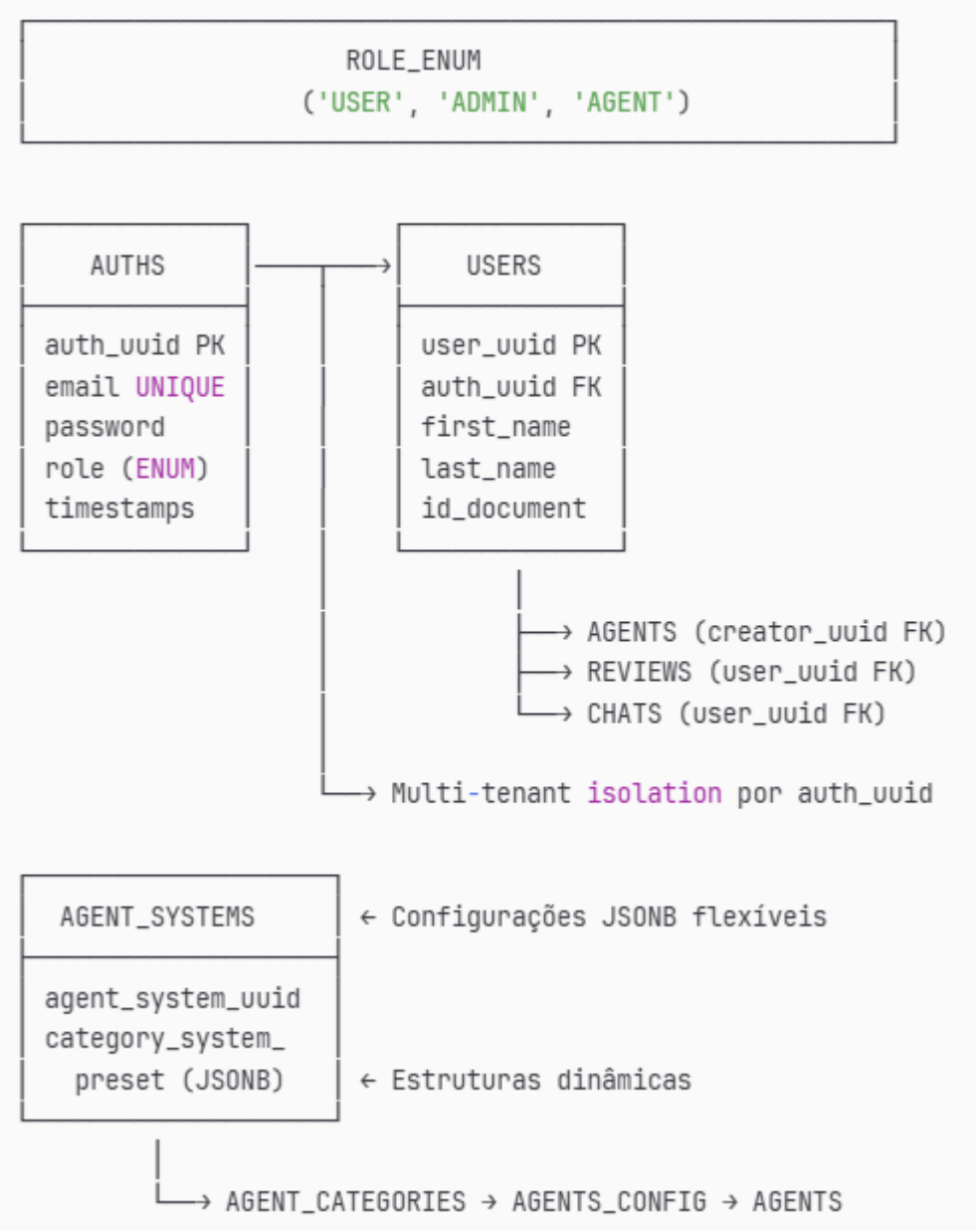
```
├── cmd/
│   └── main.go                  # Inicializador
│
├── internal/
│   ├── common/                  # Módulo Compartilhada
│   │   ├── atoms/               # Átomos
│   │   │   └── common-atoms.go  # Funções auxiliares reutilizáveis
│   │   ├── db/
│   │   │   └── db-conn.go       # Conexão com banco
│   │   └── interfaces/
│   │       └── common-interfaces.go # Contratos comuns
│   │
│   └── auth-land/               # Módulos de autenticação
│       ├── auth/
│       │   ├── domain/          # Camada de Domínio
│       │   │   └── auth-domain.go
│       │   ├── repositories/    # Camada de repositório
│       │   │   └── auths-repo.go
```

```
        ├── services/          # Camada de Negócio
        │   └── auth-service.go
        ├── handlers/          # Camada de comunicação HTTP
        │   └── auth-handler.go
        ├── interfaces/
        │   └── auth-interfaces.go # Contratos de autenticação
        └── atoms/ # Átomos
            └── auth-atoms.go   # Funções auxiliares reutilizáveis

    └── auth-signature/
        └── middleware/
            └── auth-middleware.go # JWT + RBAC

└── .env.example                # Variáveis de ambiente
```

- Banco de Dados



Padrões Adotados:

- Para o frontend separamos em 3 módulos (estilos, índice e scripts) para melhor organização.
- No banco de dados utilizamos a 3ª forma normal, dados multi-tenant, configuração flexível (JSONB) e Role-Based Access Control.
- Para o backend a API base utiliza clean architecture adicionado de abordagem de átomos (fragmentos ou helpers compartilháveis entre módulos ou camadas) e um micro serviço de IA para ser consumida pela API base.

Explicações Técnicas:

O projeto foi desenvolvido separando responsabilidades e permitindo escalabilidade independente de cada componente. Composta por três macro-camadas principais: interface do usuário (Frontend), camada de aplicação (Backend) e camada de dados (Banco de Dados).

A interface foi construída com HTML, CSS e JavaScript, esta separação modular facilita a manutenção e reutilização de código.

No backend temos como API principal desenvolvido em Golang, que oferece alta performance e leveza em concorrência, simplicidade sintática e semântica, e flexibilidade de contratos. Para as IAs , foi idealizado um micro serviço em Python com Flask e LangChain para o processamento da linguagem natural e gerenciamento dos agentes de IA.

O banco de dados feito com PostgreSQL separamos em várias tabelas (Autenticação, Usuário, Agentes, Avaliações, Chats, Mensagens), criamos indices para otimização. A infraestrutura é feita multi-tenant, JSONB, Role-Based Access Control e controle de mudanças com “created_at” e “updated_at”.

1.4. Testes e Qualidade

Tipos de testes e ferramentas utilizadas:

- Testes do front feitos a partir da extensão Live Server do VS Code.
- Testes de integração (frontend ↔ backend ↔ banco).
- API: Makefile e cURL

Relatórios de Bugs:

Frontend	Quebra de imagem a depender da resolução da tela	Baixo	Resolvido
	Usuário perde login ao recarregar a página	Crítico	Resolvido
	Formulário aceita emails inválidos	Médio	Resolvido
Backend	Sem bugs registrados	—	—
Banco de dados	Sem bugs registrados	—	—

1.5. Implantação e Manutenção

Plano de implantação:

- Implantação inicial em ambiente de testes (servidor web).
- Ajustes a partir de feedback dos usuários.
- Monitoramento de desempenho e escalabilidade.
- Atualizações contínuas para novas funcionalidades.

Versionamento:

- Versionamento do projeto feito com Github

Melhorias futuras:

- Sistema de pagamento
- Diferenciação entre usuários
- Aba de comunidade

2. Considerações Finais

O projeto avançou significativamente desde a segunda parcial, tornando-se uma plataforma funcional para democratização do acesso à inteligência artificial. Os progressos alcançados demonstram a viabilidade técnica e o potencial de impacto da solução proposta. Os avanços conquistados nas sprints 3 e 4 consolidaram a base tecnológica necessária para as funcionalidades avançadas planejadas.