

# PUCNITORIAS



# PUC Minas

Gerenciamento de monitorias



# Nossa equipe



Camilla Vitoria  
Bueno da Rocha

*Developer*



Diogo Barros de  
Rezende

*Developer*



Marcus Navarro  
Gabrich

*Scrum Master/Developer*



Matheus Rangel  
de Figueiredo

*Developer*



Rodrigo de  
Oliveira Gomes

*Product Owner/Developer*



# Contexto



# PROBLEMA

É vigente a necessidade de recursos variados zelando pelo ensino acadêmico, uma das ferramentas mais importantes são as monitorias: Sistema de aulas opcionais nas quais alunos de alto desempenho dão aulas para estudantes de semestres anteriores.

Essa dinâmica mostra-se ineficiente em diferentes cenários:

- Nos dias em que alunos descumprem o combinado (por exemplo chegando atrasados);
- Nos períodos de provas, nos quais as monitorias ficam muito cheias;
- Quando o próprio monitor não vai a monitoria.



# Sobre o projeto

## Nossa missão:

Buscar garantir, por meio de um sistema de gerenciamento, uma maneira mais eficaz de organizar as monitorias, com o intuito de que todos os que participam delas, tenham acesso às informações e materiais das mesmas, facilitando a vida, dos monitores e dos alunos, com tabela de horário, e marcações de aulas.

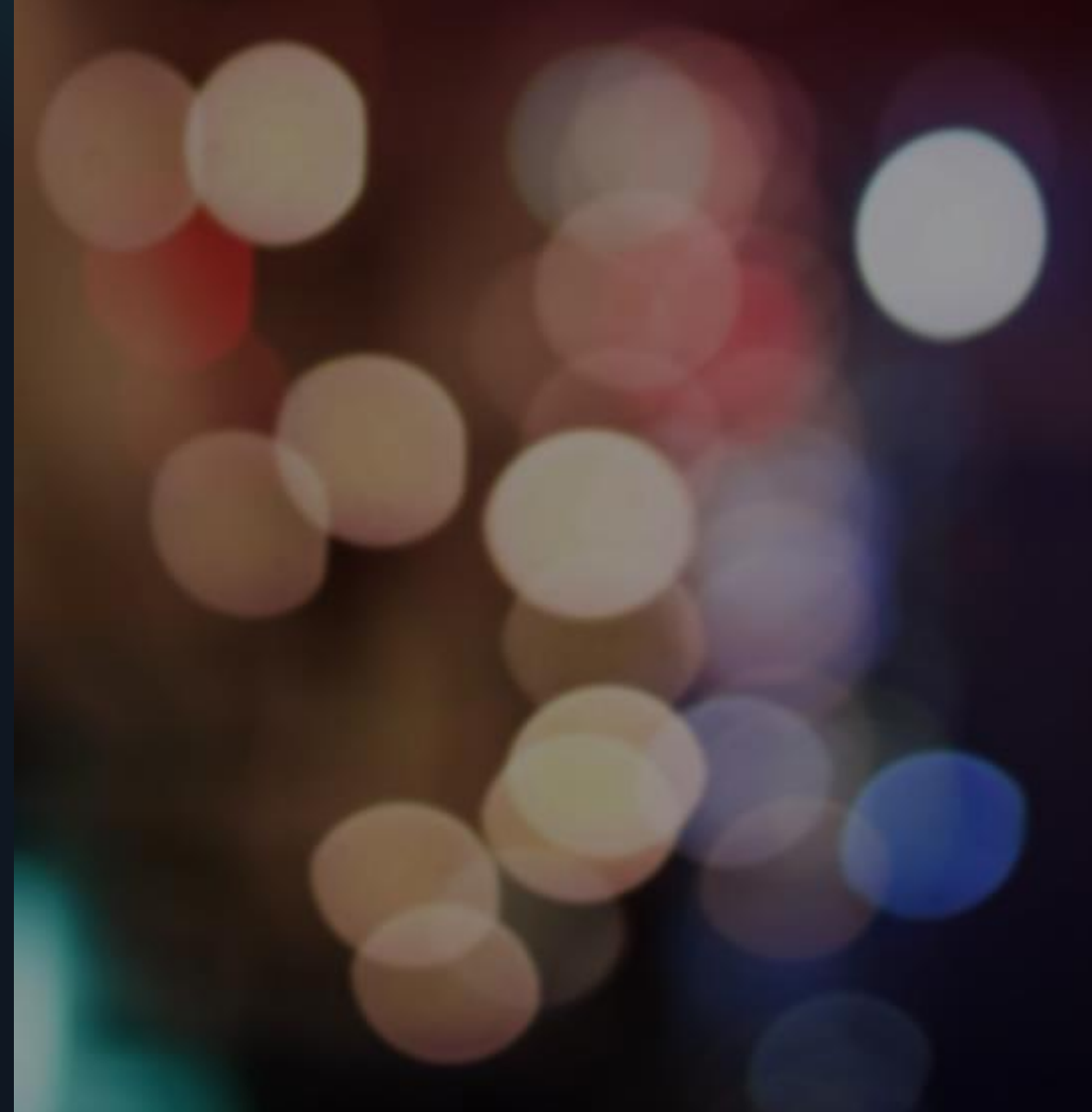




# SOLUÇÃO

Criar uma plataforma que seja rápida, fácil e eficiente em solucionar:

- Problemas na disseminação de informação;
- Problemas nos horários;
- E problemas na qualidade das monitorias.





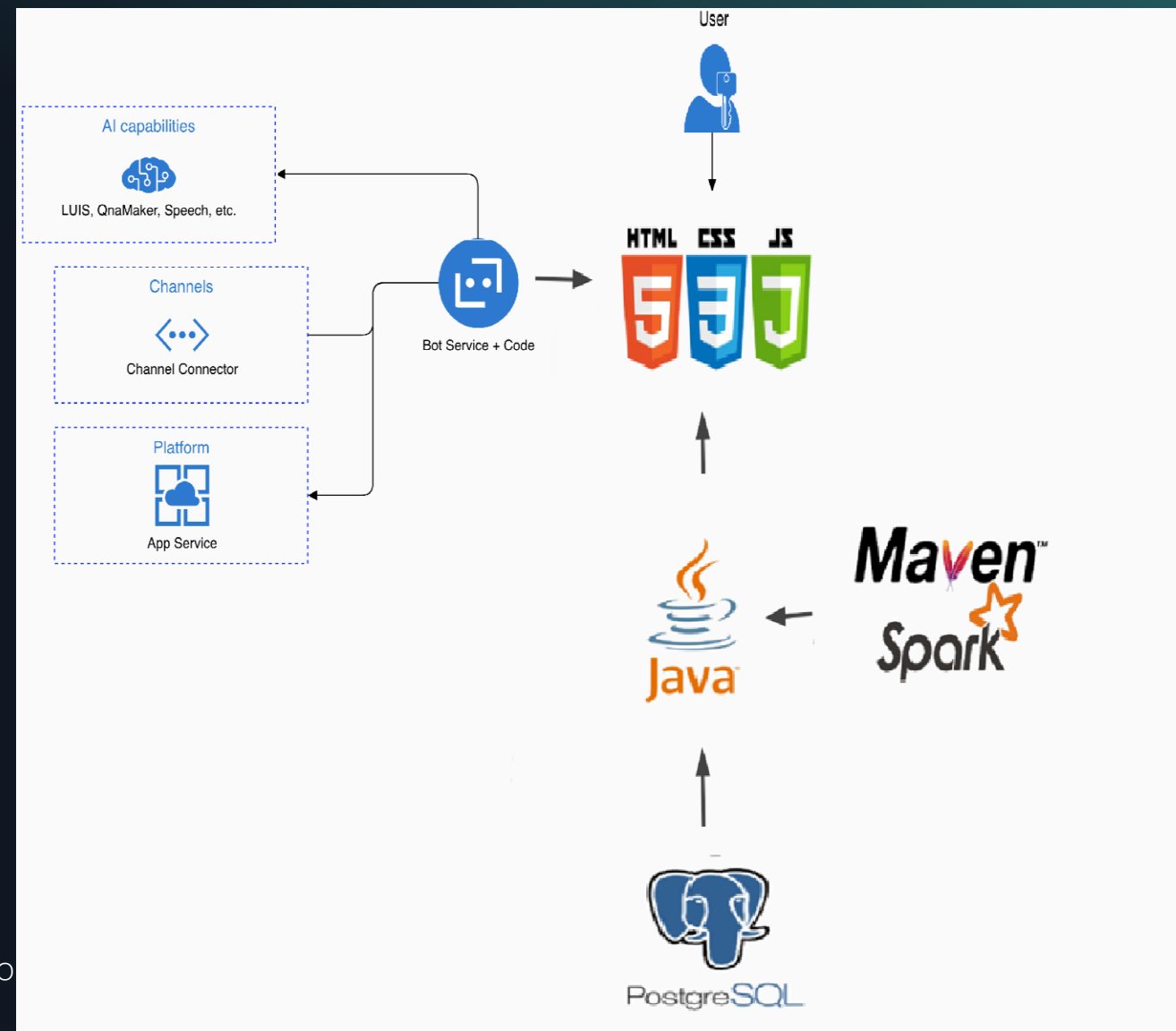


# Metodologia de trabalho



# Tecnologias

- Front-End: O grupo utiliza as tecnologias de HTML, CSS E JavaScript;
- Back-End: Na formação do back-end é utilizado java e JavaScript como linguagem de programação principal em conjunto com os frameworks Maven e Spark;
- Banco de Dados: para o modelo da aplicação do DB esta sendo utilizado o SGBD relacional postgresSQL.
- Sistemas Inteligentes: Na aplicação de Sistemas Inteligentes utiliza-se como host os serviços do Azure com um código próprio desenvolvido através de type script com frameworks SDK 4.0.

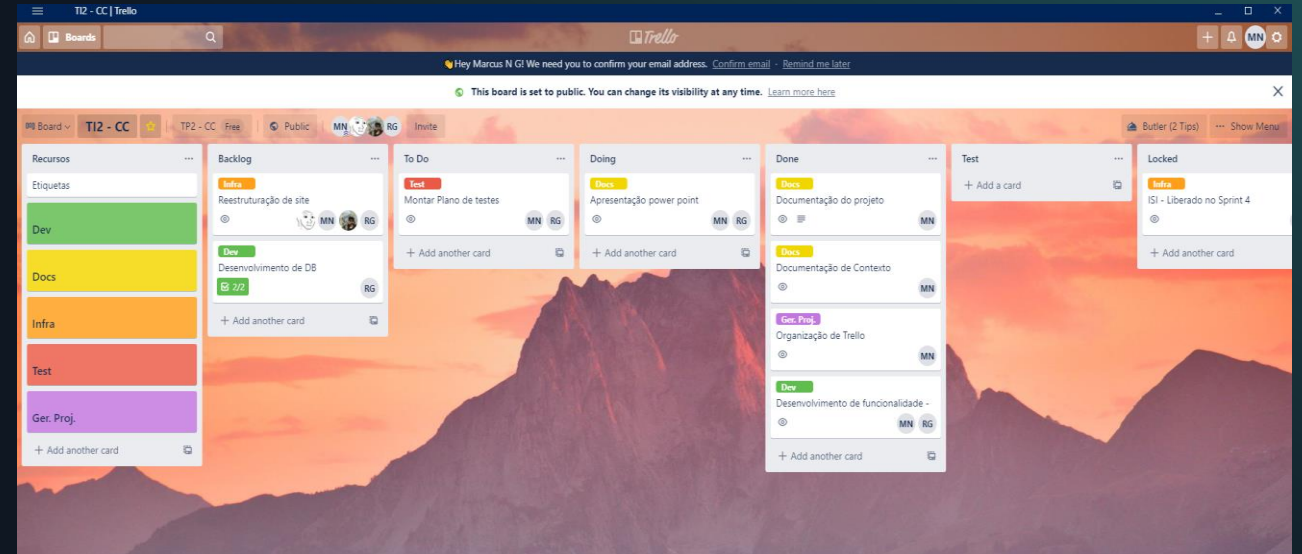




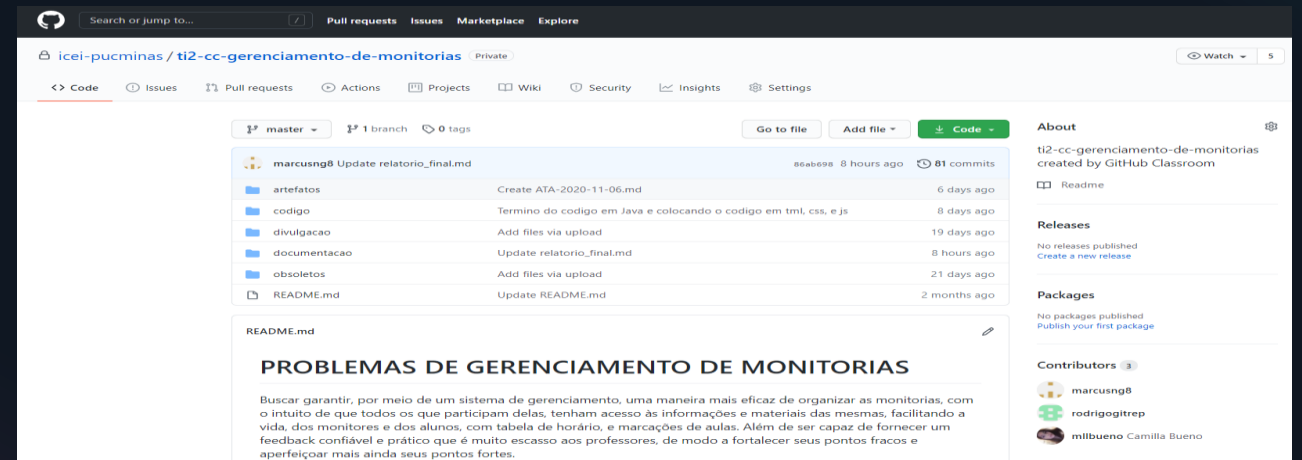


# Gerenciamento do projeto

## Estrutura Kambam: Trello / Github projects



## Versionamento de Código: GitHub hist




# Trello

The screenshot displays a Trello workspace with a board named "T12 - CC". The interface includes a top navigation bar with a home icon, a "Boards" tab, a search bar, and the Trello logo. A notification banner at the top right states: "Hey Marcus N G! We need you to confirm your email address. [Confirm email](#) · [Remind me later](#)". Below this, a message indicates: "This board is set to public. You can change its visibility at any time. [Learn more here](#)".

The board is organized into several columns, each with a title and a list of cards:

- Recursos**: A sidebar on the left containing a search bar "Etiquetas" and five color-coded category buttons: "Dev" (green), "Docs" (yellow), "Infra" (orange), "Test" (red), and "Ger. Proj." (purple). It also includes a "+ Add another card" button.
- Backlog**: Contains two cards. The first is labeled "Infra" and titled "Reestruturação de site", with assignees MN and RG. The second is labeled "Dev" and titled "Desenvolvimento de DB", with a progress indicator "2/2" and assignee RG. It includes a "+ Add another card" button.
- To Do**: Contains one card labeled "Test" titled "Montar Plano de testes", with assignees MN and RG. It includes a "+ Add another card" button.
- Doing**: Contains one card labeled "Docs" titled "Apresentação power point", with assignees MN and RG. It includes a "+ Add another card" button.
- Done**: Contains three cards. The first is labeled "Docs" titled "Documentação do projeto" with assignee MN. The second is labeled "Docs" titled "Documentação de Contexto" with assignee MN. The third is labeled "Ger. Proj." titled "Organização de Trello" with assignee MN. It also includes a card labeled "Dev" titled "Desenvolvimento de funcionalidade -" with assignees MN and RG. It includes a "+ Add another card" button.
- Test**: Contains a "+ Add a card" button.
- Locked**: Contains one card labeled "Infra" titled "ISI - Liberado no Sprint 4" with assignee MN. It includes a "+ Add another card" button.

The background of the board features a scenic image of a mountain range under a sunset sky. The bottom right corner of the board interface shows a "Butler (2 Tips)" button and a "Show Menu" button.










[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

[icei-pucminas / ti2-cc-gerenciamento-de-monitorias](#) Private Watch 5

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

master 1 branch 0 tags

[Go to file](#) [Add file](#) [Code](#)

	marcusng8 Update relatorio_final.md	86ab698 8 hours ago 81 commits
	artefatos Create ATA-2020-11-06.md	6 days ago
	codigo Termino do codigo em Java e colocando o codigo em tml, css, e js	8 days ago
	divulgacao Add files via upload	19 days ago
	documentacao Update relatorio_final.md	8 hours ago
	obsoletos Add files via upload	21 days ago
	README.md Update README.md	2 months ago

README.md

## PROBLEMAS DE GERENCIAMENTO DE MONITORIAS

Buscar garantir, por meio de um sistema de gerenciamento, uma maneira mais eficaz de organizar as monitorias, com o intuito de que todos os que participam delas, tenham acesso às informações e materiais das mesmas, facilitando a vida, dos monitores e dos alunos, com tabela de horário, e marcações de aulas. Além de ser capaz de fornecer um feedback confiável e prático que é muito escasso aos professores, de modo a fortalecer seus pontos fracos e aperfeiçoar mais ainda seus pontos fortes.

### About

ti2-cc-gerenciamento-de-monitorias created by GitHub Classroom

[Readme](#)


### Releases


No releases published  
[Create a new release](#)


### Packages

No packages published  
[Publish your first package](#)

### Contributors 3

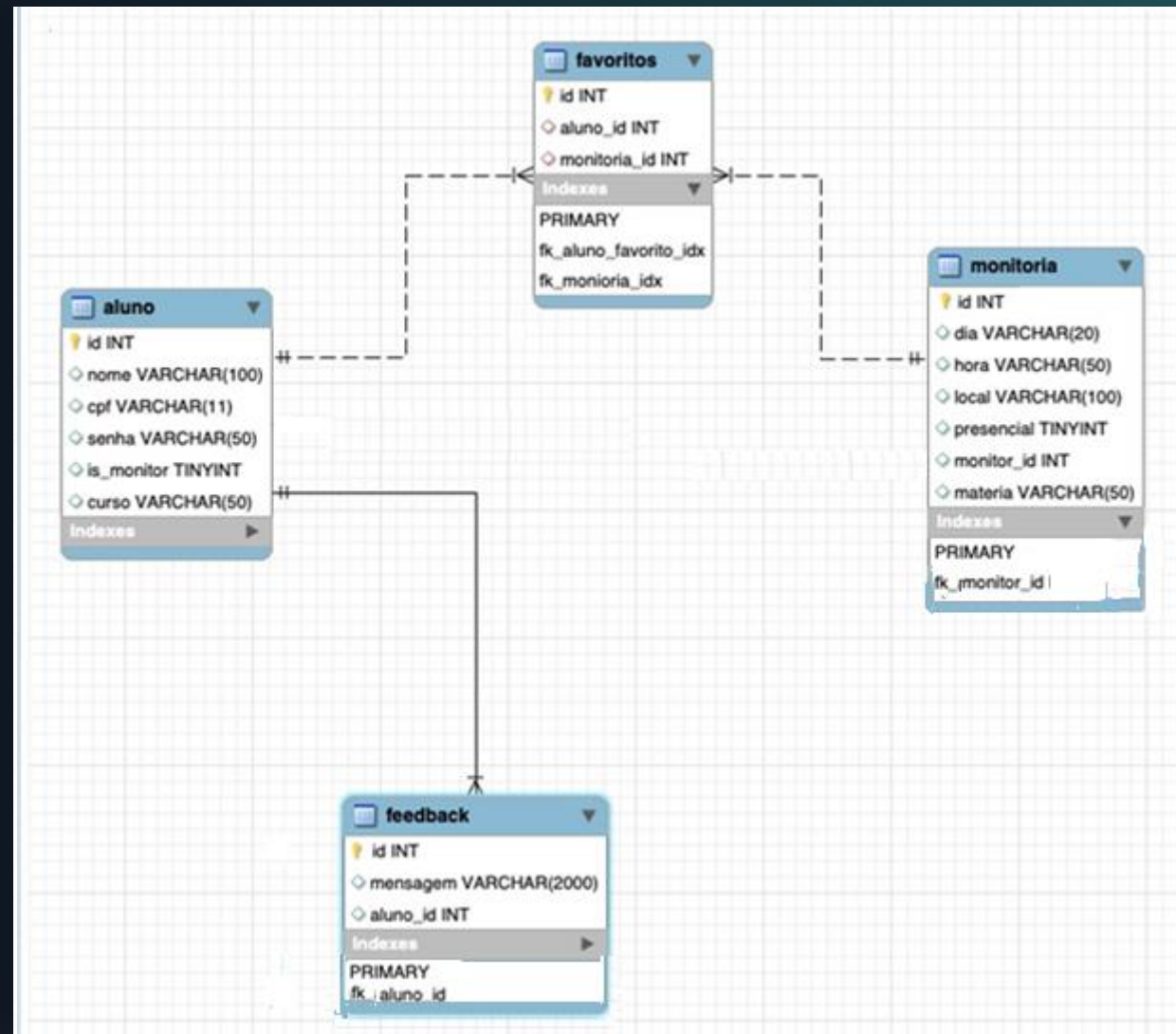
 marcusng8

 rodrigogitrep

 mllbueno Camilla Bueno




# Modelagem de dados








# Hospedagem

# Banco de Dados


**DATA**


 Datastores >  postgresql-tetrahedral-

SERVICE heroku-postgresql    PLAN hobby-dev    BILLING APP  teste-bd-gerenciamento

Overview    Durability    **Settings**    Dataclips

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database.

Please note that **these credentials are not permanent**.

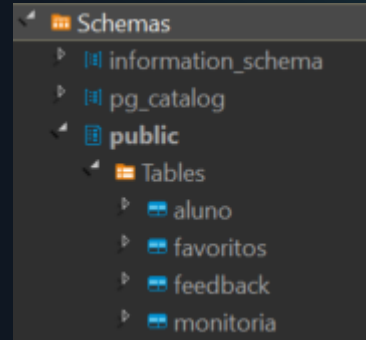
Heroku rotates credentials periodically and updates applications where this database is attached.

Host	
Database	
User	
Port	
Password	
URI	
Heroku CLI	





# Implementação BD



```
CREATE TABLE 'aluno' (
  'id' INT NOT NULL,
  'nome' VARCHAR(100) NOT NULL,
  'cpf' CHAR(11) NOT NULL UNIQUE,
  'senha' VARCHAR(50) NOT NULL,
  'is_monitor' INT NOT NULL,
  'curso' VARCHAR(50) NOT NULL,
  PRIMARY KEY ('id')
);
```

```
CREATE TABLE 'monitoria' (
  'id' INT NOT NULL,
  'dia' VARCHAR(20) NOT NULL,
  'hora' VARCHAR(50) NOT NULL,
  'presencial' BOOL NOT NULL,
  'monitor_id' INT NOT NULL,
  'local_monitoria' VARCHAR(50),
  'materia' VARCHAR(50) NOT NULL,
  PRIMARY KEY ('id'),
  CONSTRAINT 'fk_aluno'
    FOREIGN KEY ('monitor_id')
      REFERENCES 'aluno' ('id')
);
```

```
CREATE TABLE 'favoritos' (
  'id' INT NOT NULL,
  'aluno_id' INT NOT NULL,
  'monitoria_id' INT NOT NULL,
  PRIMARY KEY ('id'),
  CONSTRAINT 'fk_aluno'
    FOREIGN KEY ('aluno_id')
      REFERENCES 'aluno' ('id'),
  CONSTRAINT 'fk_monitoria'
    FOREIGN KEY ('monitoria_id')
      REFERENCES 'monitoria' ('id')
  ON DELETE CASCADE
);
```

```
CREATE TABLE 'feedback' (
  'id' INT NOT NULL,
  'mensagem' VARCHAR(2000) NOT NULL,
  'aluno_id' INT NOT NULL,
  PRIMARY KEY ('id'),
  CONSTRAINT 'fk_aluno'
    FOREIGN KEY ('aluno_id')
      REFERENCES 'aluno' ('id')
);
```



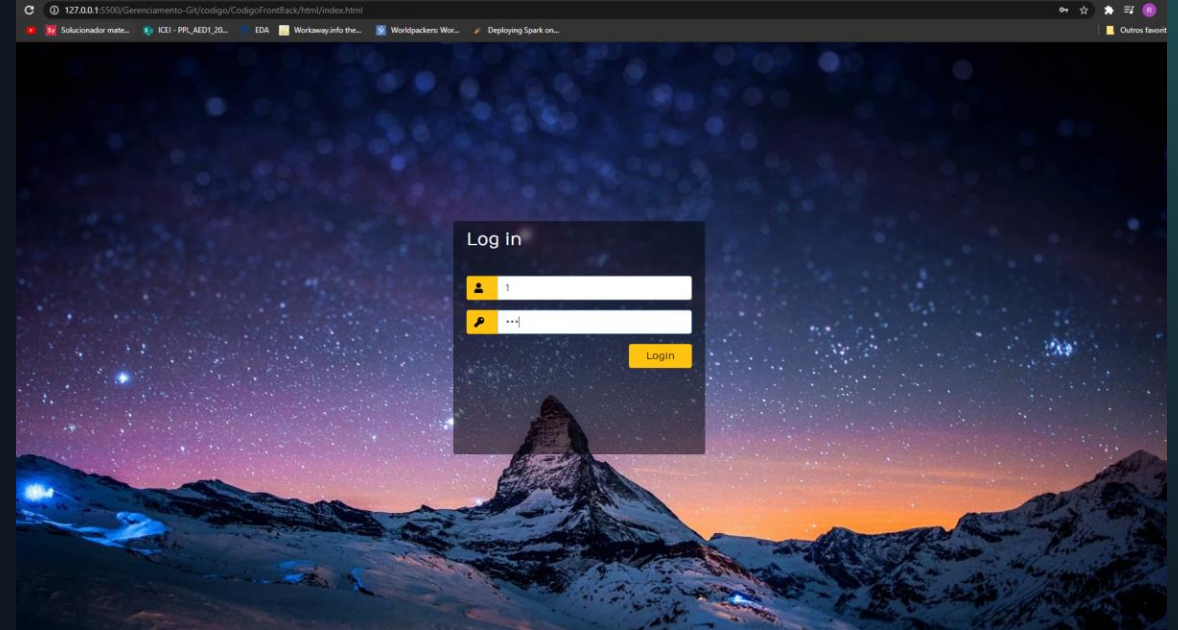
# Implementação Front-End integrado ao Back-end em Java

```
function obterDadosAJAX(){
    //executar chamada AJAX para a API do JSONSERVER
    let xhr = new XMLHttpRequest();
    xhr.onload = verificaLogin;
    xhr.open('GET', 'http://localhost:3214/aluno/getAll');
    xhr.send();
}

function verificaLogin(dados){
    dados = JSON.parse(this.responseText);

    for(i = 0; i < dados.length; i++){
        let nomeAluno = dados[i].nome;
        let idAluno = dados[i].id;
        let tipoAluno = dados[i].is_monitor;
        let senhaAluno = dados[i].senha;
        if(nome.value == idAluno){
            i = dados.length;

            if(senha.value == senhaAluno){
                login(idAluno, nomeAluno, tipoAluno);
            }else alert("Usuário ou senha incorretos!");
        }
    }
}
```



```
//CONEXOES DO ALUNO
```

```
get("/aluno/getAll", (request, response) -> {
    response.header("Content-Type", "application/json");
    response.header("Content-Encoding", "UTF-8");
    return alService.getAllAlunoLogs();
});
```

```
//FIM CONEXOES DO ALUNO
```



# Implementação do Back-End integrado ao Banco de Dados

```
▼ app
  > Main.java
▼ dao
  > CredenciaisDB.java
  > DAOAlunos.java
  > DAOFavoritos.java
  > DAOFeedback.java
  > DAOMonitoria.java
▼ model
  > Aluno.java
  > Favoritos.java
  > Feedback.java
  > JsonFormatter.java
  > Monitoria.java
▼ services
  > AlunoService.java
  > FavoritosService.java
  > FeedBackService.java
  > MonitoriaService.java
```

```
public Object getAllAlunoLogs() {
    StringBuffer returnValue = new StringBuffer("[");
    conexao.conectar();
    if(conexao.getAll() != null) {
        Aluno[] a = conexao.getAll();
        for (int i = 0; i < a.length; i++) {
            if(i != a.length-1)
                returnValue.append(a[i].toJson()+"");
            else
                returnValue.append(a[i].toJson());
        }
        returnValue.append("]");
        conexao.close();
        return returnValue.toString();
    }
}
```



- ▼ app
  - > Main.java
- ▼ > dao
  - > > CredenciaisDB.java
  - > DAOAlunos.java
  - > DAOFavoritos.java
  - > DAOFeedback.java
  - > DAOMonitoria.java
- ▼ model
  - > Aluno.java
  - > Favoritos.java
  - > Feedback.java
  - > JsonFormatter.java
  - > Monitoria.java
- ▼ services
  - > AlunoService.java
  - > FavoritosService.java
  - > FeedBackService.java
  - > MonitoriaService.java

```
public Aluno[] getAll(){
    Aluno[] alunos = null;

    try {
        Statement st = conexao.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_ONLY);
        ResultSet rs = st.executeQuery("SELECT * FROM aluno;");
        if(rs.next()){
            rs.last();
            alunos = new Aluno[rs.getRow()];
            rs.beforeFirst();

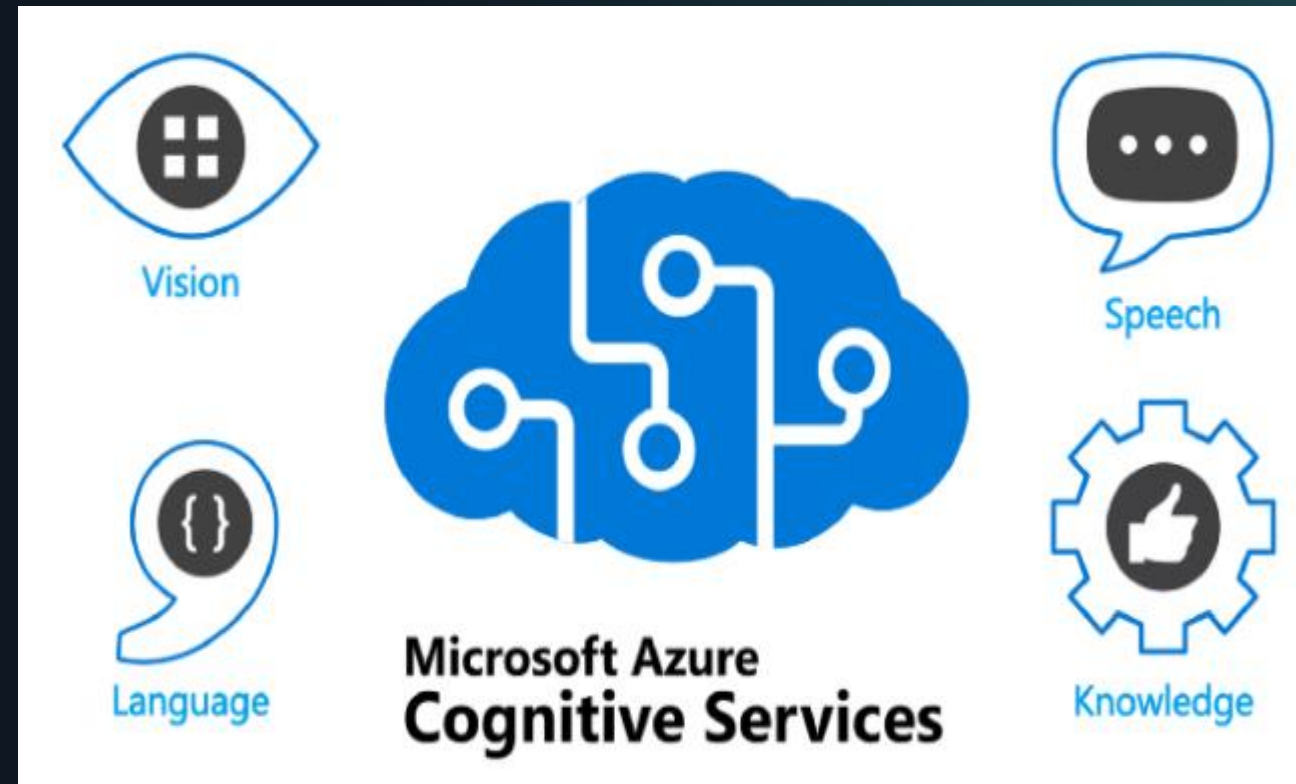
            for(int i = 0; rs.next(); i++) {
                alunos[i] = new Aluno(rs.getInt("id"), rs.getString("nome"), rs.getString("cpf"), rs.getString("senha"),
                                     rs.getInt("is_monitor"), rs.getString("curso"));
            }
        }
        st.close();
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
    return alunos;
}
```

```
public class Aluno implements JsonFormatter{
    private int id;
    private String nome;
    private String cpf;
    private String senha;
    private int is_monitor;
    private String curso;
```



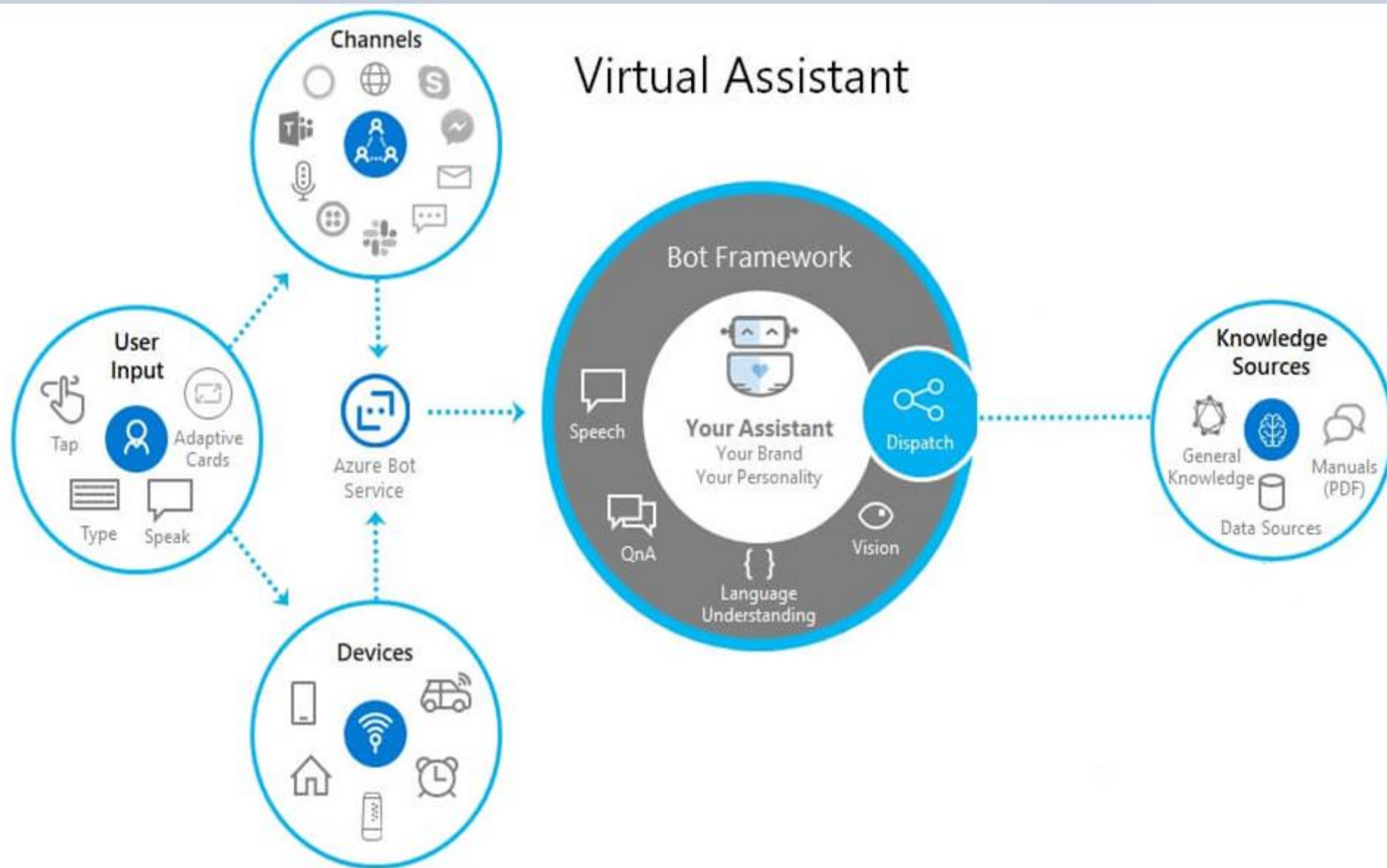
# Especificação dos Elementos de Sistemas Inteligentes

- Ira ser utilizado o Microsoft Cognitive Services. Afim de processamento de linguagem natural.
- Será utilizado o Azure Bot Services e todas suas extensões baixáveis como botbuilder. Afim do desenvolvimento do Sistema do chatbot.
- Sera utilizado o typescript afim de facilitar o reconhecimento de códigos pelo Sistema da azure.





# Virtual Assistant



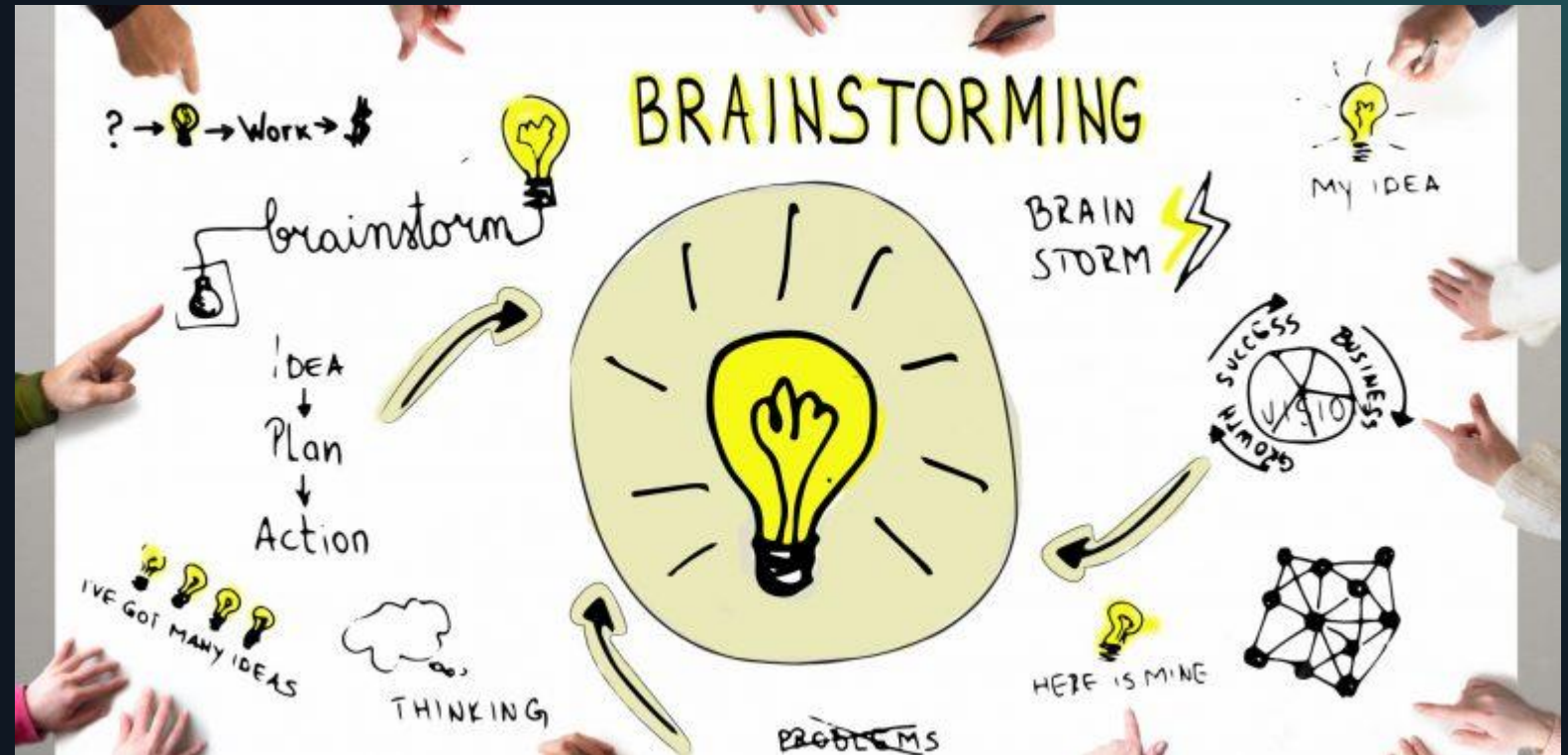


# Futuras implementações





- Pesquisas de monitorias por filtro;
- Optimização Front-End;
- Implementação de Chatbot iterativo;
- Analise de implementação Cross plataforma através do Microsoft Azure.



**PucNITORIAS**



**PUC Minas**

**Obrigado!**