**Universidade Federal de Ouro Preto** **UFOP**
Instituto de Ciências Exatas e Biológicas ICEB
Departamento de Computação DECOM

Trabalho Prático:

# Saiki

## Engenharia de Software II - BCC323

**Andrei Miguel Cristeli**
**Cristiano Augusto Dias Mafuz**
**Gabriel Carlos Silva**
**Hebert Luiz Madeira Pascoal**
**Pedro Parentoni**
**Victor Emmanuel Susko Guimarães**
**Victor Xavier Costa**

Ouro Preto,
23 de agosto de 2025.

# Conteúdo



Versão 1.0

### Resumo

Este documento objetiva a apresentação das especificidades do desenvolvimento do sistema para um jogo de adivinhação sobre temas da Ciência da Computação (Algoritmos, Personalidades, Linguagens de programação e afins). O propósito geral é reforçar de maneira prática os conceitos relativos à área de Engenharia de Software, apresentando o planejamento, organização, e progresso da equipe durante o desenvolvimento do projeto.


# I Introdução

Neste trabalho, apresenta-se uma visão geral sobre o processo de desenvolvimento do projeto *Saiki* para a disciplina de Engenharia de Software. Especificamente, apresentaremos detalhes sobre a organização e distribuição das tarefas entre a equipe e as tecnologias utilizadas.

# II Tecnologias Utilizadas

As tecnologias do Saiki são, para frontend HTML5 e CSS, e para o backend e banco de dados o Django. O Django é um framework web em Python que permite criar sites e sistemas, oferecendo ferramentas prontas para lidar com banco de dados, segurança e interface administrativa. Além disso, na confecção dos testes, foi utilizado o Selenium, que constitui uma ferramenta usada para automação de navegadores web.

# III Descrição do Projeto

Saiki é um jogo interativo e educativo baseado na web, desenvolvido para ensinar algoritmos clássicos de forma divertida e visual.

Os usuários podem jogar uma sessão normal ou ser desafiados a adivinhar o "algoritmo do dia" analisando dicas técnicas relacionadas a algoritmos conhecidos em ordenação e busca em grafos, como Bubble Sort, Merge Sort, Quick Sort, Dijkstra, A*, BFS, DFS e outros.

Cada palpite é avaliado comparando os atributos técnicos do algoritmo, e a interface destaca cada campo com um indicador colorido:

- **Verde:** correto
- **Vermelho:** incorreto
- **Amarelo:** parcialmente correto

O objetivo é reforçar a compreensão dos conceitos algorítmicos pelos alunos de Ciência da Computação por meio do reconhecimento de padrões e da exposição repetida. É como o Wordle, mas focado em Ciência da Computação.

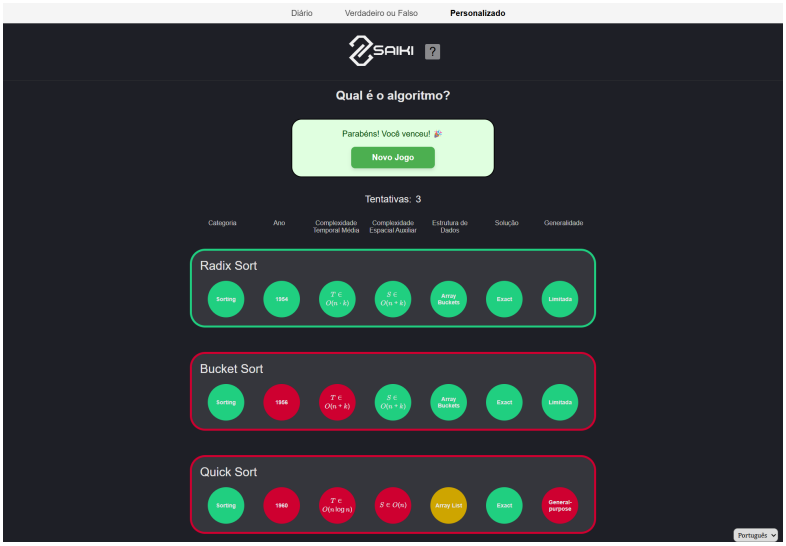


Figura 1: Exemplo de tentativas em um jogo no modo de Adivinhação.

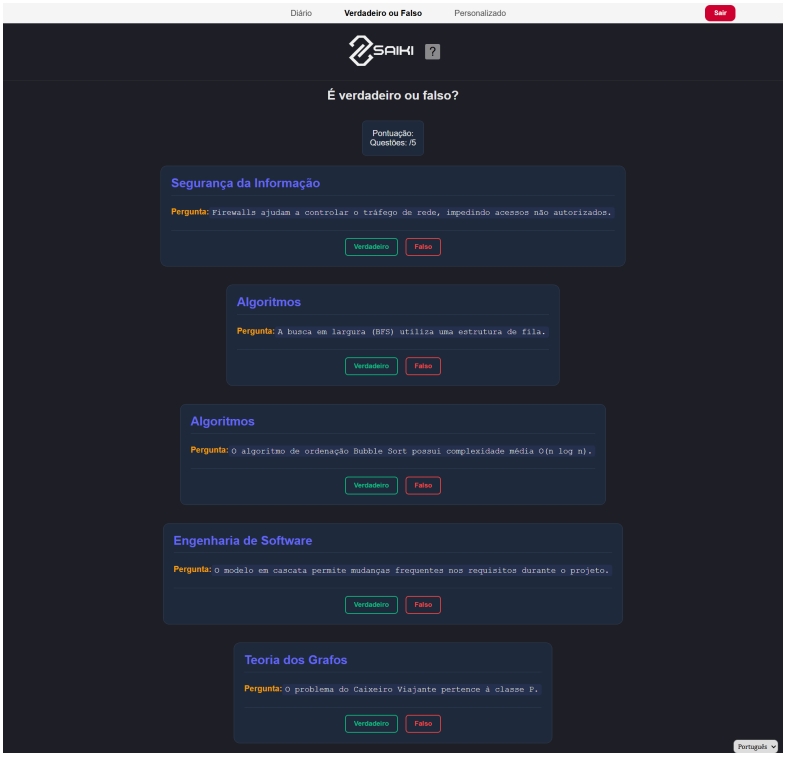


Figura 2: Exemplo de uma partida no modo Verdadeiro ou Falso.

# IV Requisitos de Projeto

## IV.I Histórias de usuários

**Jogador**

- **H01:** Eu como Jogador gostaria de Jogar o modo de adivinhação de temas da computação.
- **H02:** Eu como Jogador gostaria de jogar o modo de Verdadeiro ou Falso.
- **H03:** Eu como Jogador gostaria de compartilhar minhas estatísicas de jogo com meus amigos.
- **H04:** Eu como Jogador gostaria de visualizar sugestões de resposta baseadas no que escrevi.
- **H05:** Eu como Jogador gostaria de fazer meu cadastro de usuário no sistema.
- **H06:** Eu como Jogador gostaria de visualizar minhas estatísticas de jogo.
- **H07:** Eu como Jogador gostaria de visualizar o meu histórico de jogo.
- **H08:** Eu como Jogador Gostaria de Visualizar um manual de usuário.
- **H09:** Eu como Jogador gostaria de: Jogar o modo Multi-jogador.

**Adminstrador**

- **H10:** Eu como Administrador gostaria de gerenciar o conteúdo de cada modo de jogo.
- **H11:** Eu como Administrador gostaria de visualizar os conteúdos de cada modo de jogo.
- **H12:** Eu como Administrador gostaria de visualizar estatísticas gerais de todos os usuários.
- **H13:** Eu como Administrador gostaria de gerenciar logins.

## IV.II Casos de Uso

**UC01:** Jogador gostaria de Jogar o modo de adivinhação de temas da computação.

- **Ator:** Jogador
- **Fluxo Normal:**
  1. Visitante entra no site.
  2. Jogador acessa o modo de jogo de Adivinhação.
  3. Jogo seleciona o tópico para adivinhação.
  4. Jogador informa a tentativa.
  5. Jogo informa se o jogador venceu. Se não venceu, volta pra (4).
  6. Exibir opção de compartilhamento.
- **Extenções:**
  5a. Se Jogador sair da partida, salvar seu progresso para que seja possível retornar no mesmo estado.

**UC02:** Jogador gostaria de jogar o modo de Verdadeiro ou Falso.

- **Ator:** Jogador
- **Fluxo Normal:**
  1. Visitante entra no site.
  2. Jogador acessa o modo de jogo de Verdadeiro ou Falso.
  3. Jogo seleciona uma pergunta.
  4. Jogador informa a tentativa.
  5. Jogo informa se o jogador venceu. Se não venceu, volta pra (4).
  6. Exibir opção de compartilhamento.

## IV.III Requisitos de Banco de Dados

### Algoritmos

Cada algoritmo tem que ter informação sobre sua categoria (como ordenação, pesquisa, etc.), ano de invenção, complexidade temporal média, complexidade espacial auxiliar, estrutura-de-dados associada, tipo de solução (exata, aproximada, heurística), generalidade (se é uma solução geral, ou exige uma configuração específica da organização dos dados) e paradigma de design.

**Exemplo fictício**: o algoritmo `mínimo-por-array-ordenada` possuiria categoria *seleção*, ano *desconhecido* (básico demais para ser considerado inventado), $T \in \Theta(1)$, $S \in \Theta(1)$, estrutura-de-dados: *array*, solução do tipo *exata*, generalidade do tipo *ordenada* (já que a array precisa estar ordenada) e redução de complexidade (pois é "ótimo"em sua tarefa.)

### Estruturas de Dados

Além de algoritmo, existirão também estrutura-de-dados como entidades para serem adivinhadas no jogo. Nesse caso, cada estrutura-de-dados armazenaria informações com respeito a seu nome, sua categoria, complexidades de pesquisa, inserção e remoção, e seu ícone (imagem representativa).

**Exemplo**: Array, estrutura-de-dados linear, complexidade de pesquisa $P \in O(n)$, complexidade de inserção $I \in O(n)$ e remoção $R \in O(n)$.

$$\{\ \}$$

### Personalidades

Para cada personalidade, deve-se guardar seu nome, ano-de-nascimento, ano-de-falecimento, foto de perfil, silhueta, área de atuação, nacionalidade e prêmios (na computação). Também, a idade da personalidade — atual, se estiver vivo, ou aquela em que morreu. Uma personalidade pode ainda ser fundadora de uma área da ciência da computação.

**Exemplo**: Alan Turing, 1912 à 1954, 41 anos, atuando na área de Teoria da Computação, nacional da Inglaterra, *Smith's Prize*. (Foto ao lado.)

### Linguagens de Programação

Deverão ser armazenadas as seguintes informações para linguagens de programação: Nome, logo, ano de lançamento, paradigma, estrutura de tipo, grau de abstração e geração. Além disso, deve-se armazenar famosas aplicações em que são utilizadas.

**Exemplo**: Linguagem C, criada aproximadamente em 1970, sob o paradigma procedural, fortemente tipada, de médio nível e pertencente à terceira geração (3GL). 

### Adivinhações Diárias

As *Adivinhações Diárias* dizem respeito às entidades pré-selecionadas diariamente para serem adivinhadas globalmente, no sistema. Como idealmente a entidade é diferente de dia para dia, o histórico global das seleções devem ser armazenadas. Ou seja: todo dia uma partida "universal"é hospedada para jogo.

Sobre essas seleções diárias, deve ser possível realizar estatística, como: identificar o número de aparições da entidade selecionada, record global de tentativas, número global de jogadas, média de tempo gasto por partida, e o tempo total global gasto na partida pelos usuários.

### Partidas

Partidas representam interações de jogo (concluídas) pelos jogadores. Elas são ou do tipo *Adivinhação* ou do tipo *Verdadeiro ou Falso*. Cada tipo de partida tem um identificador único diferente. Partidas de Adivinhações referem-se ao modo "Principal"do jogo ilustrado na figura 1. Já partidas de verdadeiro ou falso são partidas em que há uma pergunta que deve ser respondida com verdadeiro ou com falso. No geral, partidas devem informar: o(s) jogador(es) participante(s) e tempo total de jogo da partida.

As partidas de adivinhação, especificamente, devem guardar: o número de tentativas, as tentativas em si (ou seja, o que cada usuário tentou em ordem, até acertar), e o tempo gasto em cada tentativa por jogador.

Já as partidas de Verdadeiro ou Falso: o texto do enunciado, a resposta para a pergunta (verdadeiro ou falso), a resposta escolhida pelo jogador. Várias perguntas de V/F serão pré-alocadas no banco de dados para o uso nos jogos.

### Sessões de Jogo

Uma parte dos sistema será o potencial suporte a partidas multi-jogadores. Para tanto, as *sessões* correspondentes devem poder ser registradas. Elas consistirão em *lobbys* (ou salas) de interações entre os jogadores.

As sessões serão informações globais do sistema vinculadas com os usuários (jogadores). Elas conterão as seguintes informações: datas de início e término, tempo da sessão, chave pública da sessão em seu momento de vida, jogadores que participaram da sessão, partidas jogadas na sessão, um resumo estatístico geral da sessão, e o chat da sessão.

Sessões conseguem hospedar inúmeras partidas. No banco de dados, só será armazenada sessões com pelo menos uma partida. E uma partida está vinculada com somente uma sessão.

### Jogador

O site terá um sistema de login. Assim sendo, informações básicas do usuário (jogador), como nome (de entrada e cadastro) e senha, deverão ser armazenados. Além disso, o nome público, e o e-mail. Cada jogar terá associado a si um número identificador único. Deve ser armazenado

também detalhes de sua última partida não terminada, para eventual retomada, e a data de último login até então. Sobre as suas informações estatísticas: deve-se armazenar o número de partidas jogadas e tentativas totais no modo adivinhação, a média de tentativas no modo adivinhação, o tempo total de jogo.

Sobre as informações sobre o último jogo (partida): quais entradas já foram inseridas, quanto tempo já decorreu, quantas tentativas se teve, e a data (e horário) da qual ele está continuando o jogo.

Deve-se armazenar também o *livro de coleção* para o jogador, em que, à medida em que uma entidade é acertada pelo usuário (uma partida é vencida), esta entidade é marcada como *colecionada*. Nesse passo, deve-se salvar a data em que ela foi primeira adicionada no livro, e o recorde: menor número de tentativas para acertar — e a data do recorde.

### Chat

Deve-se armazenar também o *chat* da sessão, como visto. O chat da sessão diz respeito às mensagens trocadas na sessão pelos jogadores. As mensagens, em codificação UTF-8, são identificadas localmente pelo seu remetente e seu id sobre o chat. Deve-se conseguir ter acesso a informação da data da mensagem. A sessão só deve armazenar o chat se ele não for vazio, durante seu tempo de vida.

## IV.IV Mapa Conceitual

O mapa conceitual de banco de dados é uma representação gráfica que mostra os principais conceitos e elementos envolvidos em um banco de dados, bem como as relações entre eles. Ele serve para organizar e visualizar de forma clara como os dados estão estruturados, facilitando o entendimento do sistema. Dessa forma, serão apresentadas as entidades e os relacionamentos do banco para o projeto proposto.

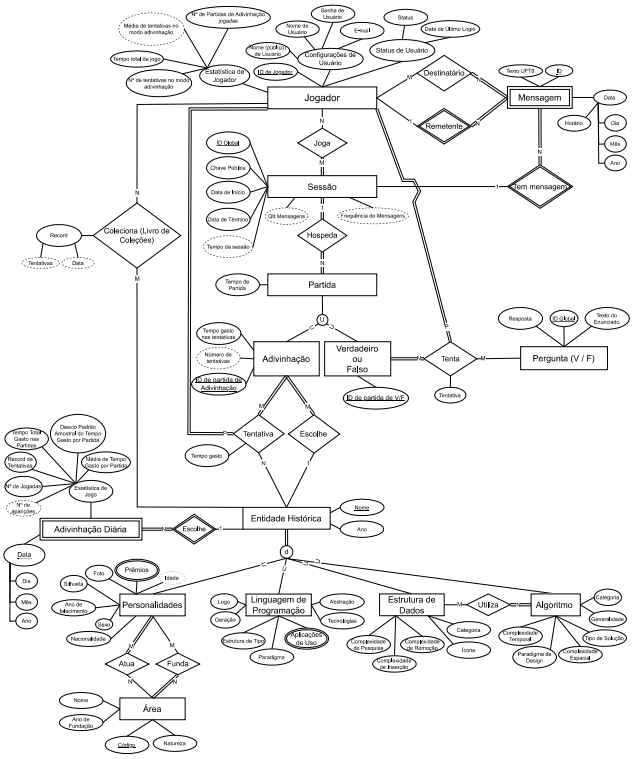


Figura 3: Mapa Conceitual do Banco de Dados.

# V Organização e distribuição de tarefas

As *issues* do trabalho não foram definidas de forma rígida, vinculando cada membro do grupo a um componente do software; isto é, nenhum integrante trabalhou em apenas uma tarefa (backend, frontend, etc). Foi convencionado dessa forma para que todos aprendessem um pouco sobre cada componente.

O projeto foi dividido em Sprints conforme as definições da metodologia Scrum. Três Sprints foram aplicadas ao longo do período de desenvolvimento, com uma média de 12 tarefas por sprint. Devido a questões de tempo, não foi possível concluir algumas funcionalidades do software, especialmente as definidas para a Sprint 3. A seguir apresenta-se a configuração final do quadro Kanban construído para o projeto. As tarefas concluídas foram divididas no quadro de acordo a sprint correspondente.

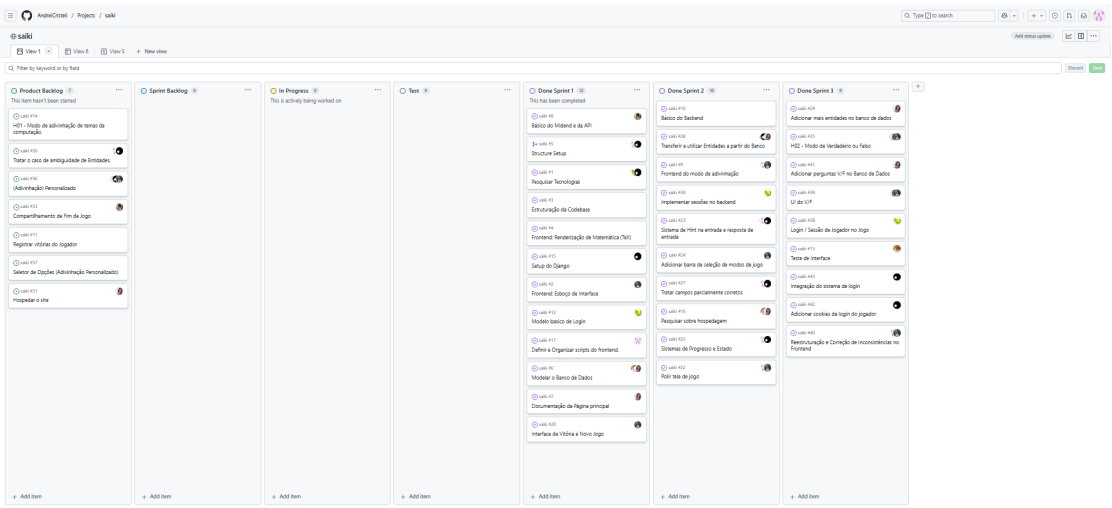


Figura 4: Configuração do quadro Kanban ao final do período de desenvolvimento

Uma visão mais detalhada sobre o progresso de cada sprint pode visualizada nas milestones que foram utilizadas para organizar as tarefas realizadas em suas respectivas sprints:

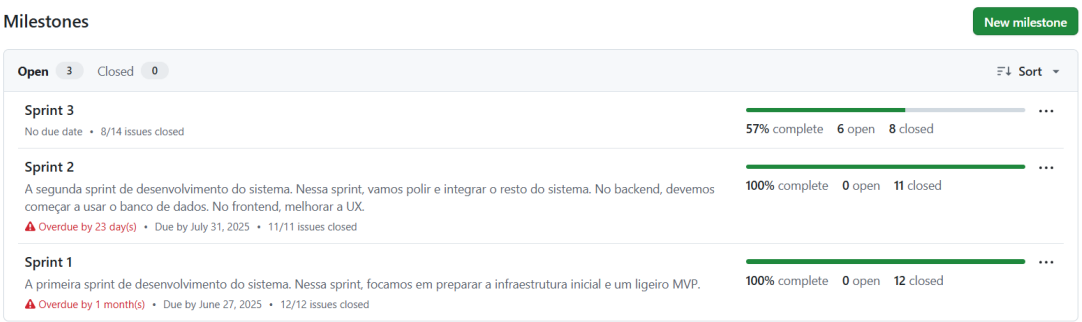


Figura 5: Progresso das milestones definidas para cada sprint ao final o período de desenvolvimento

# VI Testes Implementados

Durante o período de desenvolvimento do projeto, foram planejados testes automatizados para diferentes componentes do sistema. No entanto, a equipe teve dificuldade em encontrar tempo hábil para o desenvolvimento destes testes. Dessa forma, apenas testes de interface foram efetivamente implementados utilizando a ferramenta Selenium:

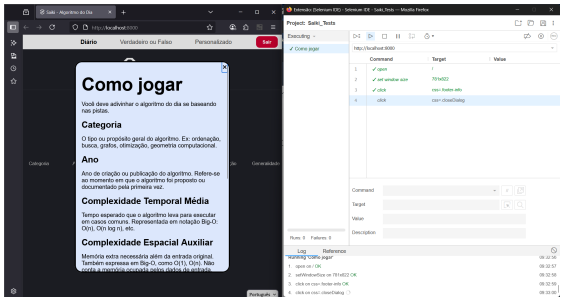


Figura 6: Teste de Interface: Opção Como Jogar

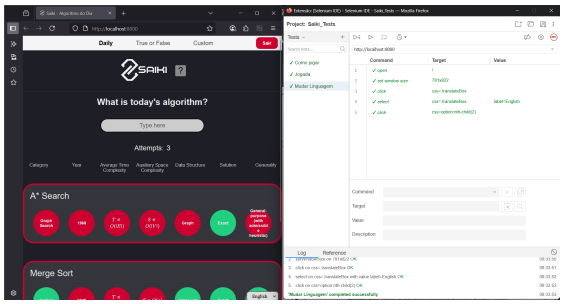


Figura 7: Teste de Interface: Alterar idioma

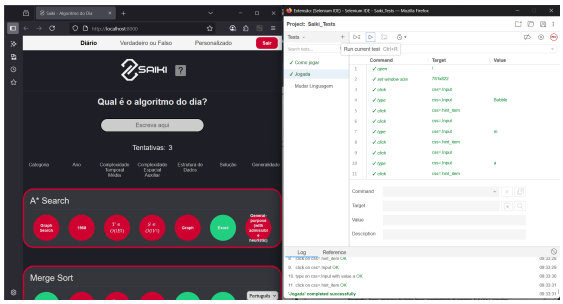


Figura 8: Teste de Interface: Simulação de tentaivas