Findoportunities.com.br

Nomes

- KAYKY MATOS SANTANA, RGM: 33345945
- LUIZ HENRIQUE SOUZA PRUDENTE, RGM:32920598
- NICOLASGUSTAVO PRADO ARLANDIS MORAES, RGM: 4933434875
- THIAGO APARECIDO GOMES, RGM:4933768391
- GUILHERME FALCK DO CARMO, RGM:4934721193
- LUCAS MATHEUS DE LIRA VASQUEZ, RGM: 33232083

Objetivo

O objetivo do projeto é criar um site de busca de emprego que conecta candidatos e recrutadores, facilitando a interação e contratação. Com recursos intuitivos para busca de vagas e gestão de candidaturas, a plataforma visa simplificar o processo de recrutamento, proporcionando uma experiência eficiente e satisfatória para ambas as partes.

Requisitos de Software

Este documento tem como objetivo descrever os requisitos funcionais, não funcionais e de negócio do sistema FindOportunities, uma plataforma de busca de emprego que conecta candidatos e recrutadores, facilitando a interação entre ambas as partes.

Escopo do Sistema

O sistema visa oferecer uma plataforma intuitiva que simplifica o processo de recrutamento para empresas e candidatos, oferecendo ferramentas para gerenciamento de vagas, perfis, candidaturas e entrevistas.

Requisitos de Negócio

- **3.1 RN001 – Aumento da Eficiência no Processo de Recrutamento**
  - O sistema deve reduzir o tempo e o esforço necessários para que recrutadores encontrem candidatos adequados e para que candidatos encontrem vagas compatíveis com seus perfis.

- **3.2 RN002 – Melhorar a Experiência do Usuário**
  - A plataforma deve proporcionar uma experiência fácil e intuitiva para recrutadores e candidatos, simplificando as ações de busca, candidatura, criação de perfis e agendamento de entrevistas.

- **3.3 RN003 – Criação de Valor para Ambas as Partes**
  - O sistema deve proporcionar uma experiência valiosa para os recrutadores, permitindo que eles encontrem candidatos qualificados rapidamente, e para os candidatos, ajudando-os a encontrar as melhores oportunidades de emprego.

Requisitos Funcionais

Funcionalidades para Recrutadores

- **RF001–Gerenciar Vagas:** O recrutador deve poder criar, editar, publicar e remover vagas de emprego.
- **RF002–Buscar Candidatos:** O recrutador deve poder buscar candidatos por meio de filtros (habilidades, experiência, localização, etc.).
- **RF003–Realizar Entrevistas:** O sistema deve permitir que o recrutador agende e conduza entrevistas com candidatos diretamente na plataforma, com integração a ferramentas de vídeo-chamada.
- **RF004–Criar Perfil de Recrutador:** O recrutador deve poder criar e gerenciar seu perfil corporativo, contendo informações da empresa, equipe de recrutamento e histórico de contratações.
- **RF005–Visualizar Informações da Conta:** O recrutador deve poder visualizar dados sobre o uso da conta, como histórico de vagas criadas, entrevistas realizadas e candidatos contatados.

Funcionalidades para Candidatos

- **RF006 – Buscar Vagas:** O candidato deve poder buscar vagas de emprego usando filtros (localização, cargo, tipo de contrato, etc.).
- **RF007 – Candidatar-se a Vagas:** O candidato deve poder se candidatar a vagas diretamente pela plataforma.
- **RF008 – Realizar Entrevistas:** O candidato deve poder agendar e participar de entrevistas via vídeo chamadas diretamente pela plataforma.
- **RF009 – Criar Perfil de Candidato:** O candidato deve poder criar um perfil contendo suas informações pessoais, experiência profissional, habilidades e currículo anexado.
- **RF010 – Visualizar Informações da Conta:** O candidato deve poder visualizar seu histórico de candidaturas e entrevistas realizadas.

Requisitos Não Funcionais

Desempenho

- **RNF001 – Tempo de Resposta:**
  - O sistema deve responder às ações do usuário (busca de vagas, filtros, carregamento de perfil) em no máximo 2 segundos, em 95% das tentativas.

Segurança

- **RNF002 – Autenticação e Autorização**
  - Todos os usuários (recrutadores e candidatos) devem ter acesso ao sistema via autenticação segura (login/senha, OAuth, etc.). Os candidatos não devem acessar informações confidenciais dos recrutadores, e vice-versa.
- **RNF003 – Armazenamento de Dados Sensíveis**
  - Dados pessoais e financeiros devem ser criptografados, seguindo as normas LGPD (Lei Geral de Proteção de Dados).

Usabilidade

- **RNF004 – Interface Intuitiva**
  - A interface do usuário deve ser intuitiva, de fácil uso, e responsiva, oferecendo uma boa experiência em dispositivos móveis e desktops.

Escalabilidade

- **RNF005 – Capacidade de Crescimento**
  - O sistema deve ser capaz de suportar um crescimento de 1000 para 100.000 usuários simultâneos sem degradação significativa de desempenho.

Confiabilidade

- **RNF006 – Disponibilidade**
  - O sistema deve estar disponível 99,5% do tempo, com janelas de manutenção limitadas a períodos fora do horário comercial.

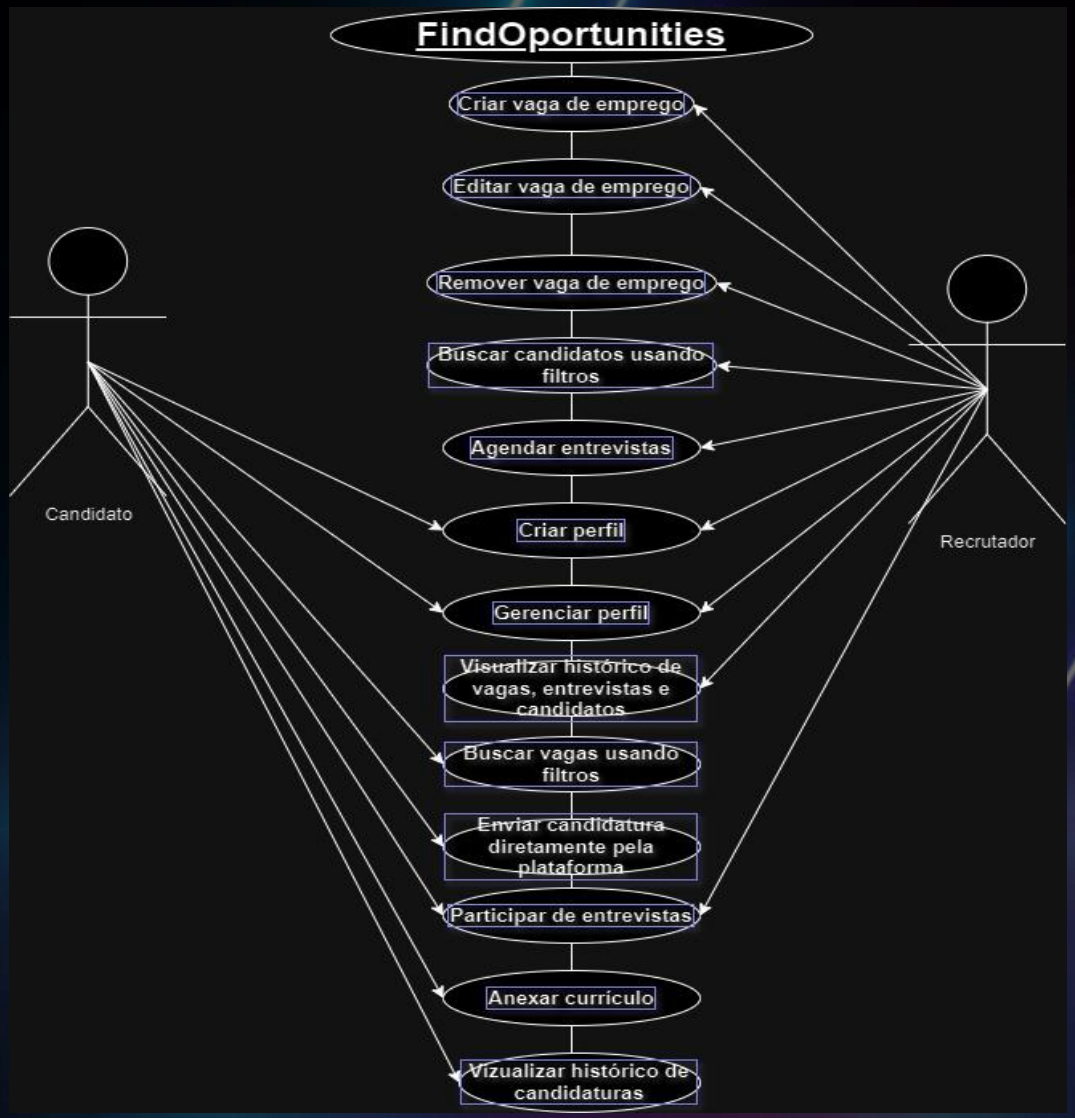
Diagramas

Diagrama de caso de uso

Diagrama de caso de uso

- **Diagrama de caso de uso**
  - Um diagrama de caso de uso é usado para representar a interação entre usuários (atores) e um sistema, descrevendo as funcionalidades principais oferecidas pelo sistema a partir da perspectiva do usuário. Ele mostra as interações entre os atores externos e os casos de uso, que representam os serviços ou funções fornecidos pelo sistema. O objetivo é fornecer uma visão clara e simplificada das funcionalidades que o sistema deve realizar, facilitando a análise e o entendimento dos requisitos de negócios.

Diagrama de caso de uso



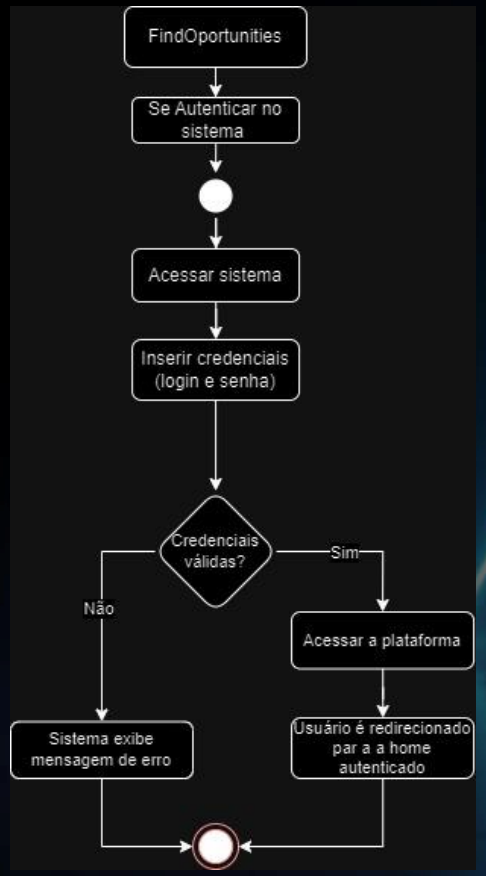
Diagramas de Atividade

Diagrama de atividades

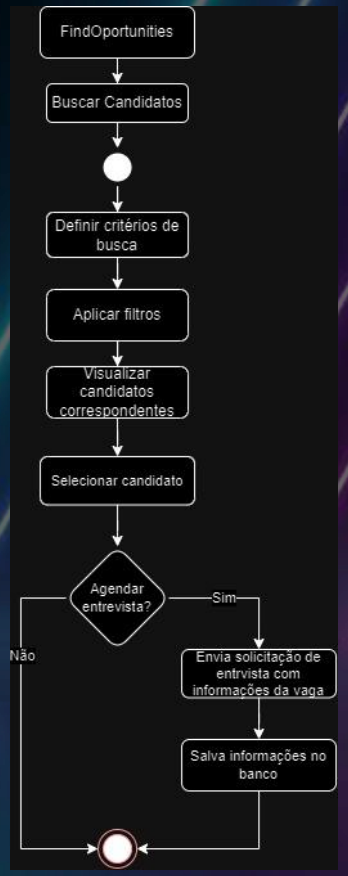
- **Diagrama de atividades**
  - Um diagrama de atividades é usado para representar o fluxo de trabalho ou a sequência de atividades em um processo. Ele mostra como as ações se conectam e como o processo flui de uma etapa para a outra, representando decisões, paralelismos e a ordem das atividades. O objetivo é ajudar a visualizar a lógica de execução de um sistema ou processo, facilitando a compreensão e o planejamento.

Diagrama de atividades

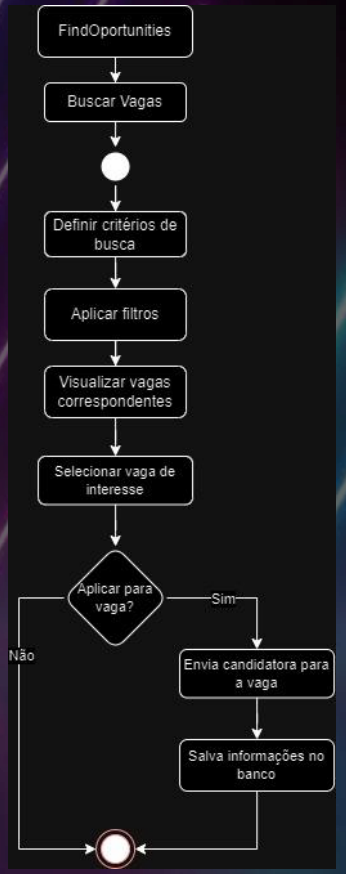
Autenticação No sistema



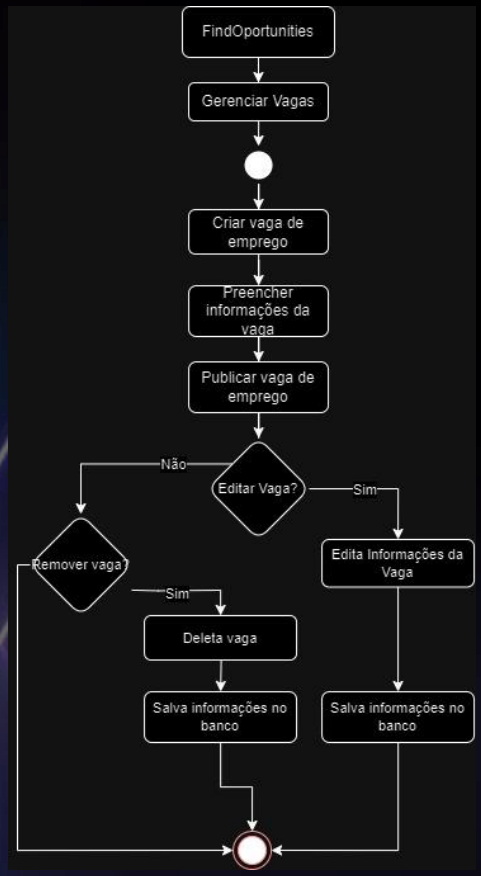
Busca por candidatos



Busca por Vagas



Gerenciamento de Vagas

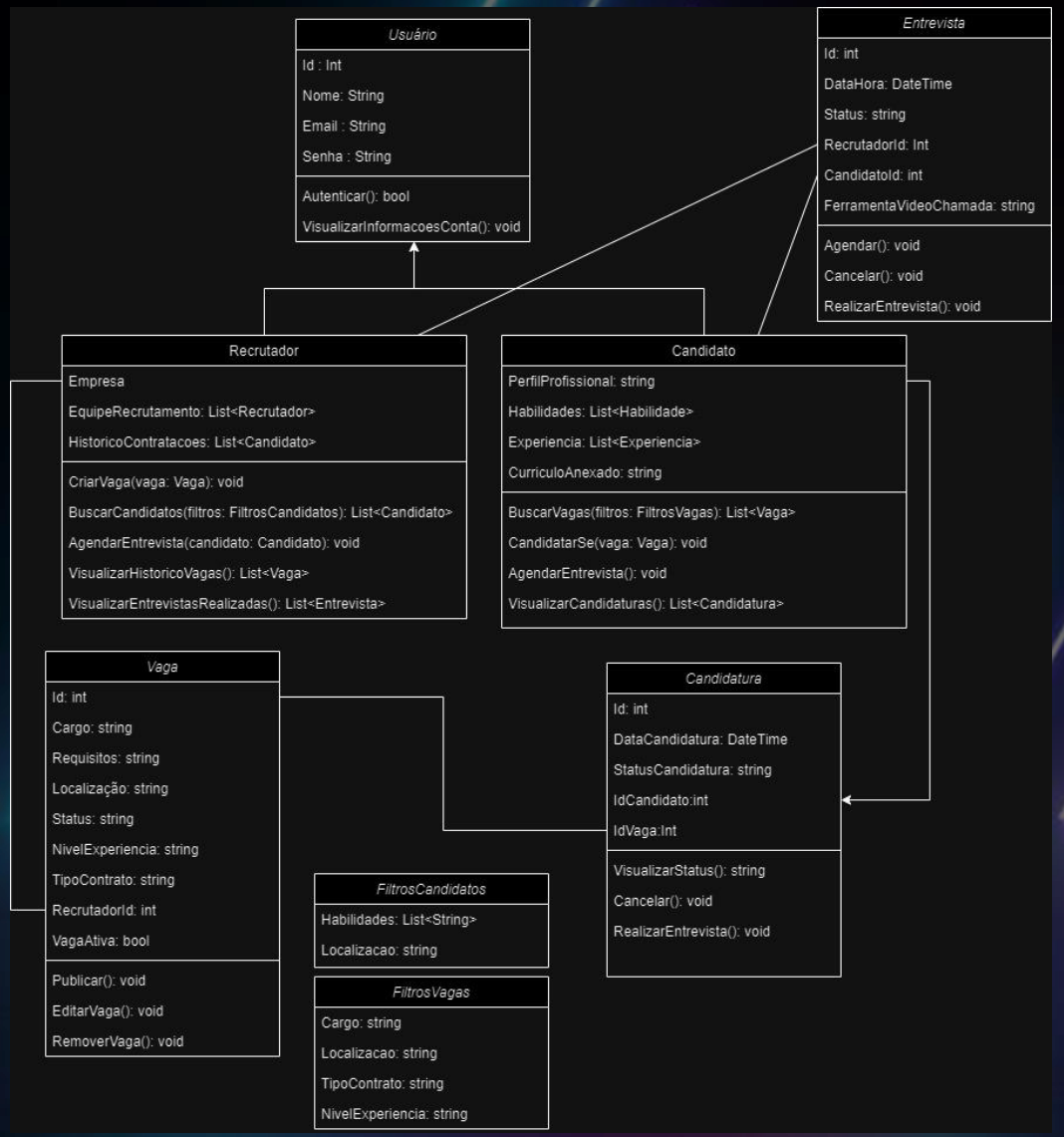


Diagramas de Classes

Diagrama de classes

- **Diagrama de classes**
  - Um diagrama de classes é usado para representar a estrutura de um sistema, mostrando as classes, seus atributos, métodos e os relacionamentos entre elas. Ele ajuda a visualizar como as diferentes partes do sistema estão conectadas e como interagem entre si. O objetivo é organizar e planejar a estrutura de um software, facilitando o design orientado a objetos, mostrando as responsabilidades de cada classe e suas associações (herança, agregação, composição, etc.).

Diagrama de classes

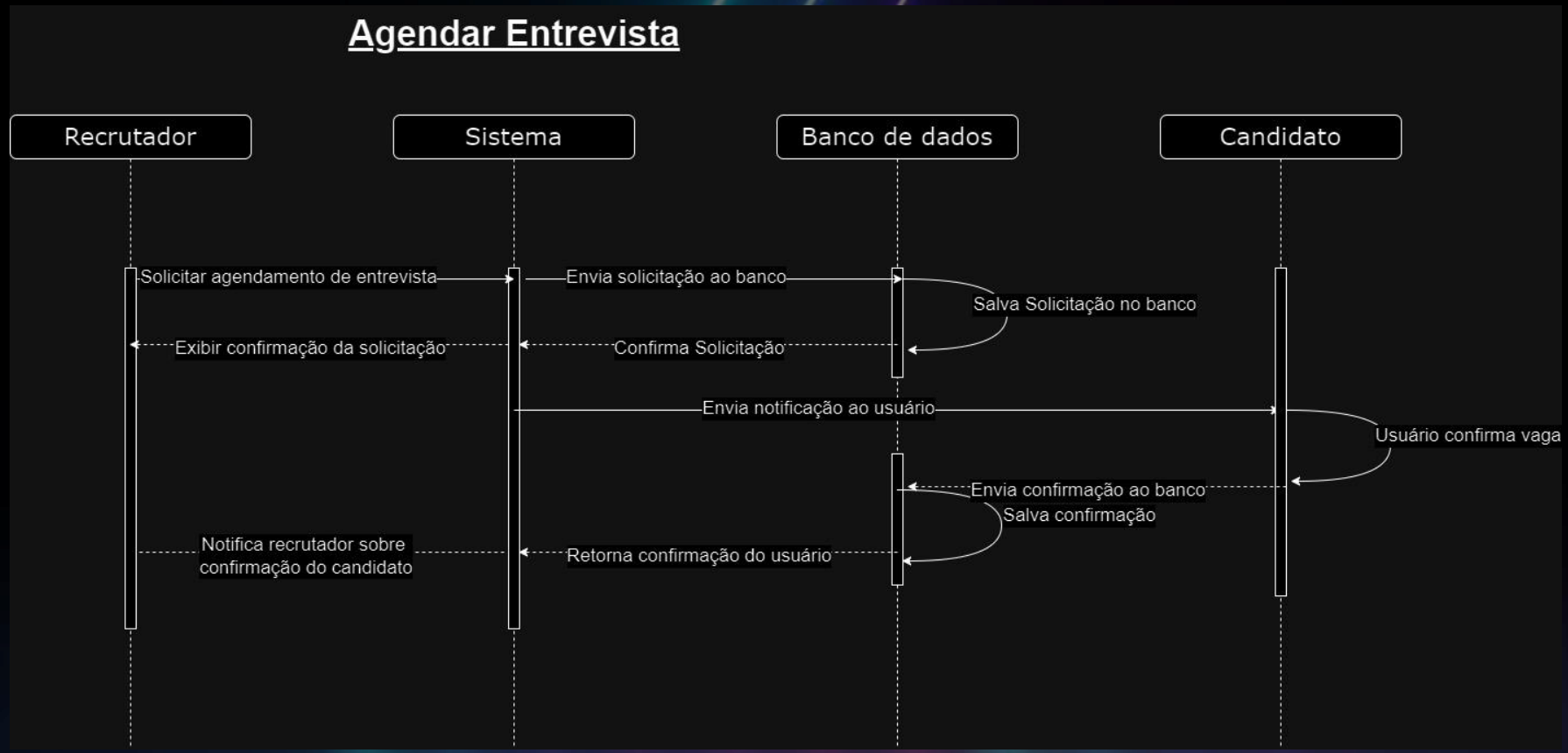


Diagramas sequenciais

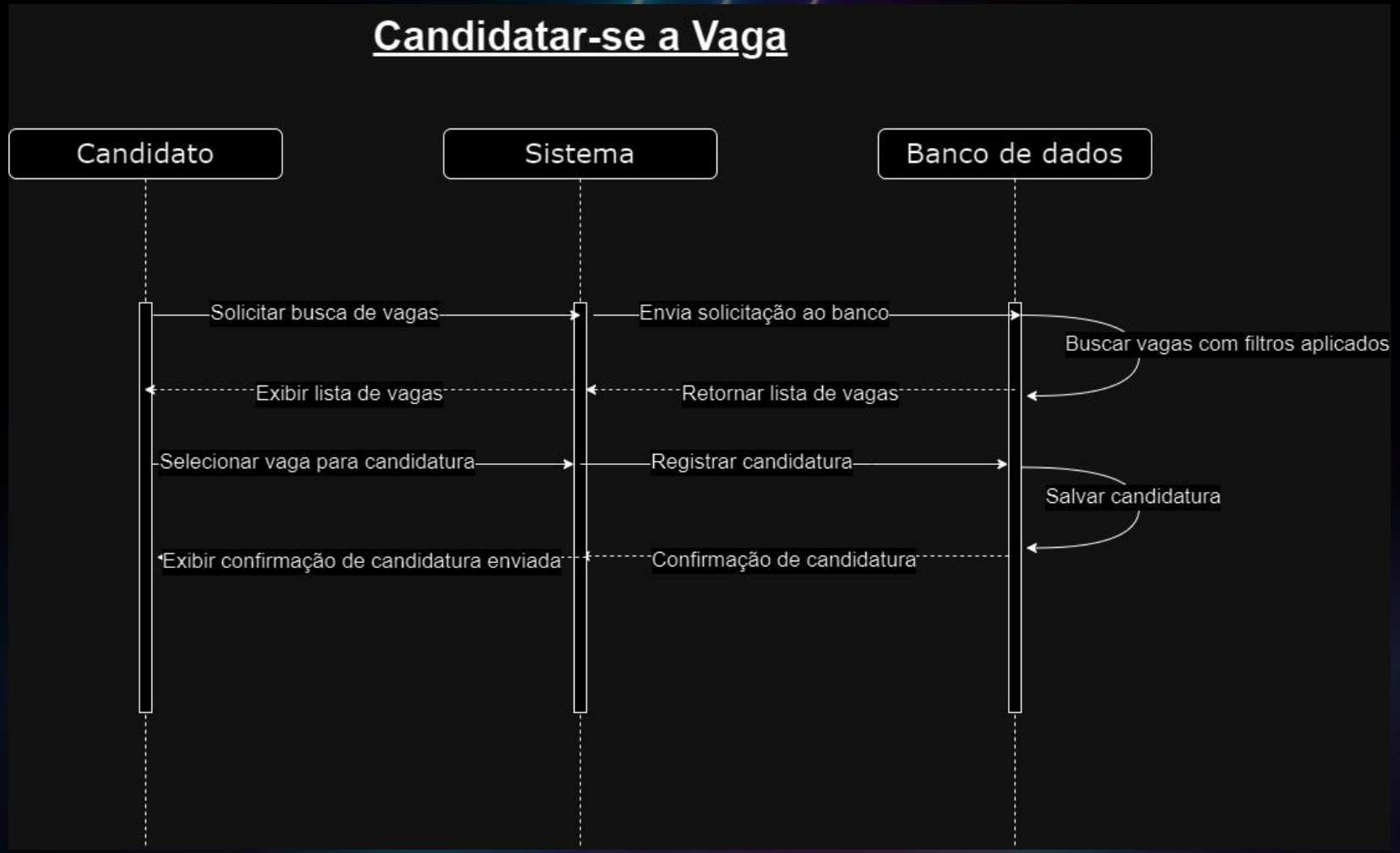
Diagramas sequenciais

- **Diagramas sequencias**
  - Um diagrama de sequência é usado para representar a interação entre os objetos de um sistema ao longo do tempo. Ele mostra a ordem em que as mensagens são trocadas entre os objetos, detalhando a sequência de chamadas de métodos ou eventos em um fluxo de execução. O objetivo é ilustrar como os componentes de um sistema colaboram para realizar uma função específica, destacando a dinâmica do sistema em termos de interação e o tempo em que essas interações ocorrem.

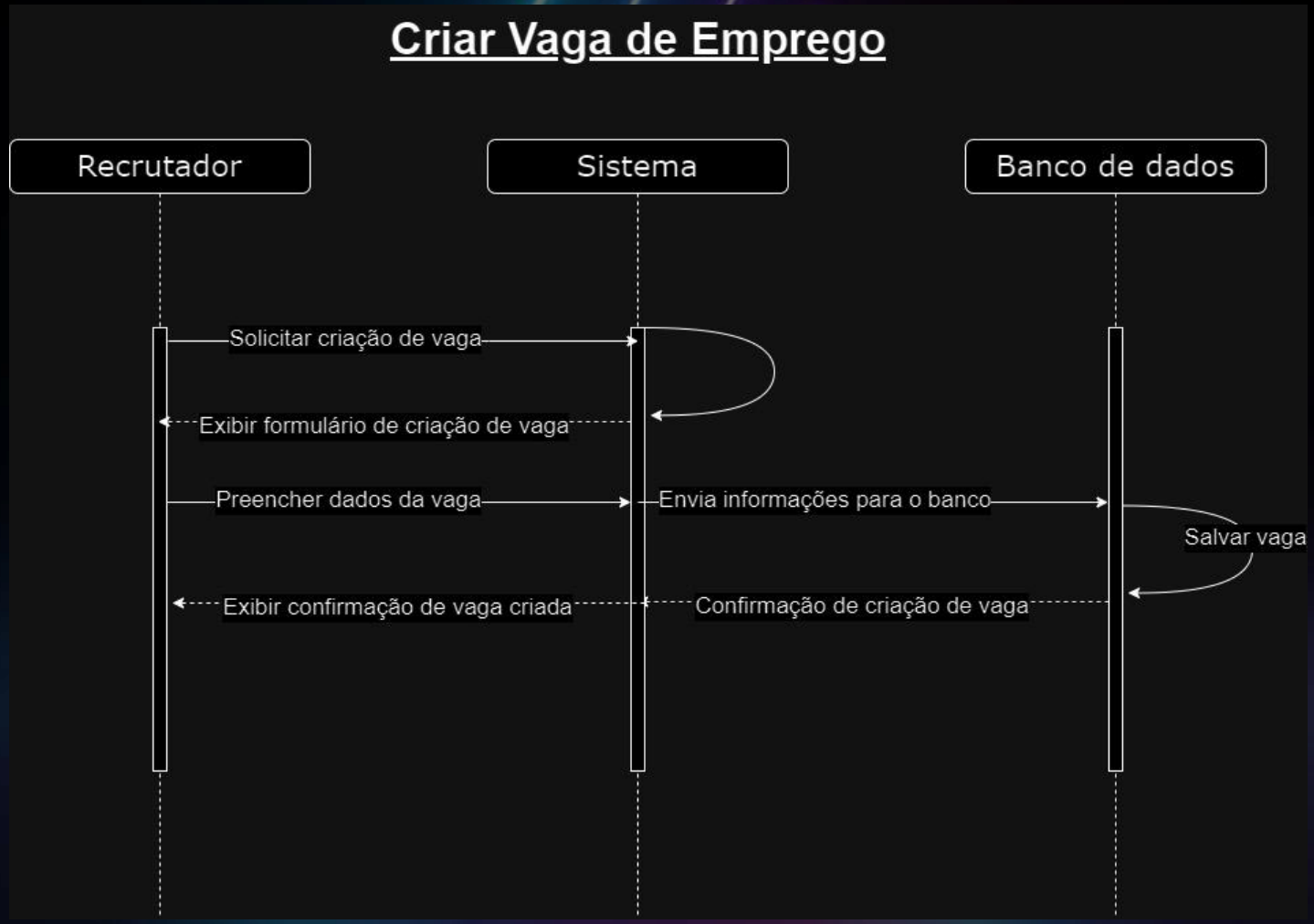
Diagramas sequenciais



Diagramas sequenciais



Diagramas sequenciais



Padrões de projeto GOF

Oque são os padrões GOF

- Os **padrões de projeto GOF** (Gang of Four) são um conjunto de 23 padrões de design orientados a objetos, documentados por quatro autores (Erich Gamma, Richard Helm, Ralph Johnson, e John Vlissides) no livro *"Design Patterns: Elements of Reusable Object-Oriented Software"*, publicado em 1994. Esses padrões fornecem soluções para problemas recorrentes no desenvolvimento de software, promovendo a reutilização de código e facilitando a manutenção e evolução do software.

Criacional - Factory Method

- **Qual problema resolve:** O Factory Method é um padrão de projeto do GoF (Gang of Four) que lida com a criação de objetos sem especificar a classe exata que será instanciada. Ele define uma interface para criar um objeto, mas permite que as subclasses decidam qual classe instanciar. Isso é útil quando você precisa instanciar objetos de diferentes classes, mas quer evitar a rigidez de depender diretamente de classes concretas.

- **Como pode ser implementado:** Neste projeto, temos a necessidade de criar diferentes tipos de perfis (recrutador e candidato), com cada um deles possuindo características específicas. Usar o Factory Method permite que o sistema crie dinamicamente o tipo correto de perfil com base no contexto (se o usuário é um recrutador ou candidato), sem precisar alterar o código sempre que um novo tipo de perfil for adicionado.

Criacional - Singleton

- **Qual problema resolve:** O Singleton Pattern garante que uma classe tenha apenas uma única instância durante toda a execução do programa, fornecendo um ponto de acesso global a essa instância. Esse padrão é particularmente útil quando precisamos de um controle centralizado de certos recursos ou informações, evitando a criação de múltiplas instâncias que possam causar inconsistências no estado do sistema.

- **Como pode ser implementado:** Ao implementar um Singleton para gerenciamento de sessão, sempre que um recrutador ou candidato fizer login, o sistema usaria essa instância única para controlar o acesso e garantir que as informações do usuário estejam corretas e seguras, evitando problemas de duplicidade ou inconsistência de dados.

Criacional - Builder

- **Qual problema resolve:** O Builder Pattern resolve o problema de criar instâncias complexas de objetos de maneira mais controlada e clara. Em vez de usar um construtor com muitos parâmetros ou uma série de chamadas de métodos setters, o Builder Pattern permite a criação de objetos passo a passo. Isso é especialmente útil quando o objeto a ser criado tem diversas configurações ou subcomponentes.

- **Como pode ser implementado:** No projeto FindOpportunities, o Builder Pattern pode ser utilizado para criar perfis de usuários (recrutadores e candidatos) de maneira flexível e extensível. Dado que os perfis podem ter muitos atributos (informações pessoais, experiência profissional, habilidades, etc.), o uso do Builder Pattern pode tornar o processo de criação e configuração desses perfis mais organizado e legível.

Comportamental - Observer

- **Qual problema resolve:** O padrão Observer permite que objetos "observadores" se inscrevam para serem notificados automaticamente sempre que o objeto central muda. O sujeito não precisa saber detalhes dos observadores, apenas que eles existem. Assim, ele apenas "avisa" os observadores quando ocorre uma mudança, permitindo que cada um deles reaja como necessário.

•Como pode ser implementado: O padrão Observer é ideal para implementar funcionalidades como notificações em tempo real, onde um objeto (observador) é notificado quando o estado de outro objeto (observado) muda. No contexto do FindOportunities, o padrão Observer pode ser utilizado para:

- **Notificações de Novas Vagas:** Quando uma nova vaga é criada ou atualizada, os candidatos que se encaixam no perfil podem ser notificados.
- **Notificações de Candidaturas:** Quando um candidato se candidata a uma vaga, o recrutador pode ser notificado.
- **Notificações de Entrevistas:** Quando uma entrevista é agendada ou cancelada, tanto o candidato quanto o recrutador podem ser notificados.

Comportamental - Strategy

- **Qual problema resolve:** O Strategy é um padrão de projeto que permite escolher diferentes algoritmos enquanto o programa está rodando. Ele organiza vários algoritmos, isolando cada um deles, e possibilita trocar um pelo outro facilmente. Isso permite mudar o comportamento de uma funcionalidade sem precisar alterar o código que a utiliza.

- **Como pode ser implementado:** No sistema FindOportunities, há funcionalidades como a busca de vagas e de candidatos, que podem ter diferentes formas de filtragem (por localização, habilidades, experiência, etc.). Usar o padrão Strategy permite que essas diferentes estratégias de busca sejam aplicadas dinamicamente, dependendo das preferências do usuário ou das necessidades do recrutador.

Comportamental -Template Method

- **Qual problema resolve:** O Template Method Pattern define o esqueleto de um algoritmo em uma classe base, deixando que as subclasses implementem os passos específicos. Esse padrão permite que parte do comportamento seja reutilizado, enquanto a implementação detalhada de certos passos pode ser personalizada pelas subclasses, promovendo flexibilidade e reutilização de código.

- **Como pode ser implementado:** Por exemplo, a classe base pode definir o esqueleto do processo de criação de perfil, enquanto subclasses para Candidato e Recrutador implementam as etapas específicas. Isso permitiria ao sistema gerenciar a criação de perfis de maneira eficiente e consistente, evitando duplicação de código, mas ao mesmo tempo oferecendo flexibilidade para lidar com as particularidades de cada tipo de usuário.

Estrutural - Facade

- **Qual problema resolve:** O Facade Pattern resolve o problema da complexidade de sistemas que possuem várias interações internas entre classes e subsistemas. Ele oferece uma interface simples e unificada para realizar operações que envolvem múltiplos componentes, escondendo a lógica interna dos clientes. Com isso, o padrão facilita o uso, reduz o acoplamento entre módulos e torna o sistema mais fácil de manter e evoluir.

- **Como pode ser implementado:** No FindOportunities, o Facade pode ser implementado para simplificar operações que envolvem várias etapas, como o agendamento de entrevistas. Atualmente, agendar uma entrevista envolve verificar a disponibilidade de candidatos e recrutadores, reservar um horário e enviar notificações. O Facade poderia encapsular toda essa lógica, expondo um método único para agendar entrevistas, tornando o processo mais simples para os usuários, tanto recrutadores quanto candidatos, e reduzindo a necessidade de interagir diretamente com as classes internas do sistema.

Estrutural - Proxy

- **Qual problema resolve:** O padrão Proxy é utilizado para controlar o acesso a um objeto, agindo como um intermediário que oferece uma interface para outro objeto. Ele resolve problemas como controle de acesso, adição de segurança, cache ou inicialização preguiçosa de objetos que são caros em termos de recursos, como objetos que demoram para carregar ou que consomem muita memória. Além disso, ele permite adicionar funcionalidades extras ao acessar o objeto sem modificar seu código diretamente.

- **Como pode ser implementado:** No sistema FindOportunities, o padrão Proxy poderia ser utilizado para controlar o acesso a dados sensíveis de candidatos e recrutadores. Um proxy de segurança poderia garantir que somente usuários autenticados acessem perfis e informações confidenciais, de acordo com o requisito de segurança (RNF002). Além disso, um proxy de cache poderia ser implementado na funcionalidade de busca de vagas, reduzindo o tempo de resposta ao evitar consultas repetidas ao banco de dados, otimizando o desempenho da aplicação conforme o requisito de tempo de resposta (RNF001).

Conclusão

- Com isso, concluímos a fase de planejamento inicial do projeto. É fundamental que, antes de iniciarmos qualquer desenvolvimento, tenhamos uma visão clara e estruturada do que será realizado. O planejamento adequado fornece a direção necessária para garantir que o projeto atinja seus objetivos de forma eficiente, evitando retrabalhos e garantindo uma execução mais alinhada às expectativas.