CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE CAMPINAS

CURSO SUPERIOR DE TECNOLOGIA EM GESTÃO DA TECNOLOGIA DA INFORMAÇÃO

HUGO TESSAROLI DA SILVA

LUIS GUSTAVO LIMA BEZERRA

**EXERCÍCIO 12 – APLICATIVO MOBILE - SIGA ALUNOS**

CAMPINAS/SP

2024-1ºsem

**SUMÁRIO**

[1. REQUISITOS 2](#_Toc17228)

[1.1 Atores 3](#_Toc26506)

[1.2 Requisitos Funcionais 3](#_Toc15027)

[1.2.1 Lançar Notas 3](#_Toc4476)

[1.2.2 Lançar Faltas 4](#_Toc11391)

[1.2.3 Lançar Datas 5](#_Toc3278)

[1.2.4 Editar/Consultar Disciplina 6](#_Toc27016)

[1.2.5 Postar Avisos 6](#_Toc25945)

[1.2.6 Lançar Documentos 7](#_Toc23322)

[1.2.7 Trocar Senha 8](#_Toc8358)

[1.2.8 Trocar Email 9](#_Toc29766)

[1.2.9 Solicitar Documentos 10](#_Toc649)

[1.2.10 Solicitar Revisão 11](#_Toc6793)

[1.3 Requisitos não Funcionais 12](#_Toc12186)

[1.4 Diagramas de caso de uso 12](#_Toc10471)

[1.5 Narrativas de Caso de Uso 12](#_Toc25713)

[1.6 Ferramentas 14](#_Toc2223)

[2. INTERFACE 14](#_Toc28628)

[1.1 Proposta de Interface 14](#_Toc30075)

[1.2 Ferramentas 16](#_Toc20274)

[2.3 Recomendações e Boas Práticas 18](#_Toc28213)

[3. GESTÃO DE CONFIGURAÇÃO 20](#_Toc2086)

[3.1 Proposta de organização de configuração 20](#_Toc8987)

[3.2 Criação de projeto e inclusão de documentos 21](#_Toc28369)

[3.3 Ferramentas 21](#_Toc28961)

[4. MÉTRICAS 22](#_Toc18510)

[4.1 Pontos de função 22](#_Toc15647)

[4.2 Métricas para desenvolvimento 22](#_Toc29076)

[4.3 Ferramentas 26](#_Toc21930)

[5. SCRUM 26](#_Toc13944)

[5.1 Backlog do Sistema 26](#_Toc30865)

[5.2 Sprints 26](#_Toc8623)

[5.3 Simulação 1° Sprint 27](#_Toc16714)

[5.4 Burndown Chart 28](#_Toc5636)

[5.5 Ferramentas 28](#_Toc12257)

[6. TESTES 29](#_Toc16927)

[6.1 Testes para o sistema 29](#_Toc13334)

[6.2 Ferramentas 31](#_Toc19338)

# REQUISITOS

## 1.1 Atores

Os atores durante a utilização do sistema podem ser resumidos em 3:

1. **Alunos**

O aluno deve conseguir consultar as informações inseridas no sistema pelos demais atores, como notas, faltas, datas de provas e avisos.

1. **Professores**

Oprofessor é o responsável por alimentar o sistema com uma série de informações conforme o andar de sua disciplina e o resultado de suas avaliações, com base nos dados citados, esse ator então, inseri diretamente no sistema, as notas e faltas dos alunos, bem como atualiza o calendário com as datas de provas e atividades vindouras, e edita os dados de sua disciplina com informações como descrição e formula para calculo de notas.

1. **Secretária**

A secretária fica responsável pelo suporte durante a utilização do sistema, seja no momento do cadastro, seja no momento de alteração de algum dado, servindo como linha de frente, para auxiliar os demais atores no caso de qualquer erro ou problema na utilização do siga

## 1.2 Requisitos Funcionais

### 1.2.1 Lançar Notas

* Descrição:

Após realização das atividades e/ou avaliações que compõem a nota da disciplina, o docente responsável pela disciplina pode lançar no sistema as notas das mesmas, por exemplo, ao aplica e corrigir uma prova, o professor pode então, lançar as notas dos alunos no sistema.

* Atores:

Professores

* Pré-Condições:

O professor e o aluno devem ter um cadastro ativo no sistema

* Pós-Condições:

As notas são lançadas e ficam disponíveis para consulta

* Fluxo Básico:

1. O professor loga no sistema
2. O professor abre a disciplina
3. Lança as notas que deseja lançar
4. O sistema calcula a nota final com base nas notas lançadas e na formula especificada pelo docente para calculo da nota

* Fluxos Alternativos:

1. Caso o aluno não esteja cadastrado, o sistema irá retornar, “Aluno não encontrado”
2. Caso o docente não esteja cadastro, o sistema ir´á retornar, “Usuário ou senha incorreta, caso esteja tendo dificuldades com o seu acesso, favor entrar em contato com o seu administrador”

### 1.2.2 Lançar Faltas

* Descrição:

Ao final da aula, o professor pode logar no sistema, abrir a disciplina, e ir a aba de aulas, encontrar a aba de faltas e selecionar a aula do dia, o sistema então irá exibir uma lista corrida de todos os alunos com matricula na disciplina naquele semestre, com todos com o campo “presente” marcado, o docente então, pode ir desmarcando o campo de acordo com a presença dos alunos, na aula em questão, e após isso, confirmar a chamada, as faltas então, são disponibilizadas na aba “Faltas Parciais”, para consulta.

* Atores:

Professores

* Pré-Condições:

1. O professor e o aluno devem ter um cadastro ativo no sistema
2. Os alunos listados devem estar matriculados na disciplina naquele semestre

* Pós-Condições:

O sistema atualiza a aba de faltas parciais

* Fluxo Básico:

1. O professor loga no sistema
2. Abre a disciplina
3. Abre a aula do dia
4. Na listagem que o sistema retornar, desmarca o campo “presente”, de acordo com a presença dos alunos na aula
5. O professor da o comando para finalizar a chamada

* Fluxos Alternativos:

1. Caso o docente não esteja cadastro, o sistema irá retornar, “Usuário ou senha incorreta, caso esteja tendo dificuldades com o seu acesso, favor entrar em contato com o seu administrador”

### 1.2.3 Lançar Datas

* Descrição:

O docente pode, dentro da disciplina, na aba de avaliações, o professor pode indicar as datas para as provas e demais atividades avaliativas.

* Atores:

Professores

* Pré-Condições:

1. A disciplina deve estar cadastrada do sistema
2. O docente deve ter acesso aquela disciplina

* Pós-Condições:

As datas são lançadas e ficam disponíveis para consulta na aba “Calendário de provas”

* Fluxo Básico:

1. O docente loga no sistema
2. O docente abre a displina
3. O docente seleciona a prova/atividade avaliativa previamente cadastrada
4. O docente atribui uma data a mesma, e da o comando ao sistema para salvar as modificações
5. As datas então, ficam disponíveis para consulta na aba “Calendário de provas”

* Fluxos Alternativos:

1. Caso a prova/atividade avaliativa a qual o docente deseja editar não esteja inclusa na disciplina, o professor pode criar, ou editar uma atividade avaliativa já existente, e então dar o comando ao sistema para salvar a alteração
2. Caso o docente não tenha acesso a disciplina, a disciplina não irá ser mostrada para ele realizar a edição das datas da mesma

### 1.2.4 Editar/Consultar Disciplina

* Descrição:

O docente pode editar, na aba “Plano de ensino”, pode selecionar as disciplinas que foram atribuídas a ele, pela gestão de sua instituição, adicionando/editando dados como descrição, ementa, datas de prova/atividades avaliativas, formula para calculo de nota final, dentre outras informações

* Atores:

Coordenador de curso, Professores

* Pré-Condições:

1. O coordenador de curso deve atribuir a disciplina ao docente
2. O docente deve ter um login válido no sistema

* Pós-Condições:

A disciplina foi editada

* Fluxo Básico:

1. A disciplina é atribuída ao docente pelo coordenador do curso
2. O docente loga no sistema
3. O docente abre a disciplina e seleciona o que deseja editar
4. O docente da o comando ao sistema para salvar as alterações

* Fluxos Alternativos:

1. Caso a disciplina não seja atribuída ao docente, ela não aparece para o docente edita-la

### 1.2.5 Postar Avisos

* Descrição:

A coordenadoria do curso, bem como a secretária, pode lançar avisos, anúncios e outros tipos de comunicados, que ficam disponíveis para visualização de todos os usuários do sistema na aba “avisos”

* Atores:

Coordenadoria dos cursos, Secretária acadêmica

* Pré-Condições:

1. Os responsáveis pelo lançamento do aviso deve ter um acesso valido ao sistema

* Pós-Condições:

1. Os avisos ficam disponíveis para consulta de todos os usuários do sistema na aba ”avisos”

* Fluxo Básico:

1. O responsáveis pelo lançamento dos avisos loga no sistema
2. O responsável entra na aba “avisos”
3. O responsável seleciona a opção “Postar aviso”
4. O responsável edita o aviso da forma que preferir
5. O responsável da o comando ao sistema para salvar as alterações
6. O aviso fica disponível para visualização de todos os usuários do sistema na aba “avisos”

* Fluxos Alternativos:

1. Caso o usuario não tenha o acesso para lançar os avisos, a opção “Postar aviso” não aparecerá para ele

### 1.2.6 Lançar Documentos

* Descrição:

Todos os usuários podem lançar e/ou consultar documentos condizentes com seus acessos, como por exemplo, a coordenadoria pode lançar as ementas, a disciplinas e os horários de aula, os professores pode lançar materiais nas disciplinas, avaliações, atividades extra-classe e materiais bibliográficos.

* Atores:

Usuários do sistema

* Pré-Condições:

1. Os usuários devem ter um login valido no sistema

* Pós-Condições:

1. O documento é lançado no sistema

* Fluxo Básico:

1. O usuário entra no sistema
2. Acessa a aba desejada
3. Seleciona a opção lançar documentos
4. Seleciona a opção de documento que deseja
5. Adiciona as opções necessárias e confirma

* Fluxos Alternativos:

1. Caso o usuário não esteja cadastro, o sistema ir´á retornar, “Usuário ou senha incorreta, caso esteja tendo dificuldades com o seu acesso, favor entrar em contato com o seu administrador”

### 1.2.7 Trocar Senha

* Descrição:

Os usuários do sistema podem trocar suas senhas de acesso, garantindo a segurança de suas contas. Este recurso está disponível na aba “configurações” do perfil do usuário.

* Atores:

Usuários do sistema

* Pré-Condições:

1. O usuário deve ter um login valido no sistema.

* Pós-Condições:

1. A senha do usuário é atualizada com sucesso.
2. O usuário recebe uma confirmação de que a senha foi trocada.
3. A nova senha deve ser usada para todos os futuros logins.

* Fluxo Básico:

1. O usuário loga no sistema.
2. O usuário entra na aba “configurações” do seu perfil.
3. O usuário seleciona a opção “Trocar senha”.
4. O usuário insere a senha atual.
5. O usuário insere a nova senha e confirma a nova senha.
6. O usuário dá o comando ao sistema para salvar as alterações.
7. O sistema confirma que a senha foi trocada com sucesso e notifica o usuário.

* Fluxos Alternativos:

1. Caso o usuário insira a senha atual incorretamente, o sistema exibe uma mensagem de erro e pede para tentar novamente.
2. Caso a nova senha e a confirmação da nova senha não coincidam, o sistema exibe uma mensagem de erro e pede para tentar novamente.
3. Caso o usuário não esteja autenticado, a opção “Trocar senha” não estará disponível e o usuário será redirecionado para a página de login.

### 1.2.8 Trocar Email

* Descrição:

Os usuários do sistema podem trocar o endereço de email associado às suas contas. Este recurso está disponível na aba “configurações” do perfil do usuário, garantindo que todas as comunicações sejam enviadas para o novo endereço de email.

* Atores:

Usuários do sistema

* Pré-Condições:

1. O usuário deveter um login valido no sistema.

* Pós-Condições:

1. O endereço de email do usuário é atualizado com sucesso.
2. O usuário recebe uma confirmação de que o email foi trocado.
3. Todas as futuras comunicações são enviadas para o novo endereço de email.

* Fluxo Básico:

1. O usuário loga no sistema.
2. O usuário entra na aba “configurações” do seu perfil.
3. O usuário seleciona a opção “Trocar email”.
4. O usuário insere o novo endereço de email.
5. O usuário confirma o novo endereço de email.
6. O usuário dá o comando ao sistema para salvar as alterações.
7. O sistema confirma que o endereço de email foi trocado com sucesso e notifica o usuário.

* Fluxos Alternativos:

1. Caso o novo endereço de email e a confirmação do novo email não coincidam, o sistema exibe uma mensagem de erro e pede para tentar novamente.
2. Caso o usuário não esteja autenticado, a opção “Trocar email” não estará disponível e o usuário será redirecionado para a página de login.

### 1.2.9 Solicitar Documentos

* Descrição:

Os alunos podem solicitar documentos específicos, como históricos escolares, declarações de matrícula e outros documentos acadêmicos. Este recurso está disponível na aba “solicitações” do perfil do aluno.

* Atores:

Usuários do sistema

* Pré-Condições:

1. O aluno deve estar autenticado no sistema.

* Pós-Condições:

1. A solicitação de documento é registrada no sistema.
2. O aluno recebe uma confirmação de que a solicitação foi realizada.
3. A solicitação é encaminhada para a secretaria acadêmica para processamento.

* Fluxo Básico:

1. O aluno loga no sistema.
2. O aluno entra na aba “solicitações” do seu perfil.
3. O aluno seleciona a opção “Solicitar documento”.
4. O aluno escolhe o tipo de documento desejado.
5. O aluno preenche as informações necessárias para a solicitação (motivo, detalhes adicionais, etc.).
6. O aluno dá o comando ao sistema para enviar a solicitação.
7. O sistema confirma que a solicitação foi registrada com sucesso e notifica o aluno.
8. A solicitação é encaminhada para a secretaria acadêmica.

* Fluxos Alternativos:

1. Caso o aluno não esteja autenticado, a opção “Solicitar documento” não estará disponível e o aluno será redirecionado para a página de login.
2. Caso as informações necessárias para a solicitação não estejam completas, o sistema exibe uma mensagem de erro e pede para o aluno preencher os dados faltantes.

### 1.2.10 Solicitar Revisão

* Descrição:

No período de até duas semanas antes do fim do semestre, os alunos podem entrar em contato com a secretaria solicitando a revisão de suas notas, a secretária então, entra em contato com a solução, e solicita ao mesmo, os critérios de avaliação e as atividades, bem como, uma nova analise dos mesmos, afim de confirmar as notas atribuídas ao estudante

* Atores:

Aluno, Secretaria, Professor

* Pré-Condições:

1. O aluno deve ter um cadastro valido
2. O aluno deve ter realizado uma solicitação

* Pós-Condições:

1. O aluno recebe um retorno acerca da sua solicitação

* Fluxo Básico:

1. O aluno loga no sistema
2. Realiza a solicitação de revisão, através da aba “Solicitar Revisão”
3. A solicitação de revisão é enviada para a secretária
4. A secretária solicita ao docente da disciplina os criterios de avaliação e as avaliações e/ou as atividades avaliativas realizadas pelo aluno, além da revisão das mesmas
5. O aluno recebe um retorno acerca de sua solicitação

* Fluxos Alternativos:

1. Se o aluno não tiver um login valido no sistema, o sistema o retornara “Usuário ou senha incorreta, caso esteja tendo dificuldades com o seu acesso, favor entrar em contato com o seu administrador”
2. Se o prazo para a solicitação tiver acabado, o sistema retornará para o aluno “O prazo para a solicitação de revisão acabou, em caso de duvidas, contate a secretária acadêmica do seu polo”

## 1.3 Requisitos não Funcionais

* Desempenho:

O sistema deve funcionar de forma otimizada e veloz, permitindo a execução de todas as funções do mesmo de forma sucinta, permitindo uma melhor troca de informações e dados entre os usuários

* Acessibilidade:

O sistema deve ser simples de se utilizar, com uma interface intuitiva, para que, todos os usuários consigam utiliza-lo da mesma forma, independente de fatores externos

* Segurança

Em vista da grande quantidade de dados dos usuários que o sistema armazena, dentre eles, documentos, senha e outros da mesma ou até de maior importância, a segurança é uma característica imprescindível para o sistema

## 1.4 Narrativas de Caso de Uso

* Lançar Notas

O professor de engenharia aplicou uma prova no dia anterior, e após corrigi-lá, deseja lançar as notas de seus alunos no sistema, o docente então, acessa o sistema, abre a disciplina a qual a prova foi aplicada, seleciona a prova previamente cadastrada e atribuindo as notas aos alunos na listagem de participantes da disciplina

* Lançar Faltas

Ao fim da aula, o professor deseja lançar as faltas de seus alunos, o mesmo então, entra no sistema, abre a sua displina e acessa a listagem de alunos, desmarcando, conforme necessidade, o campo “presente”.

* Lançar Datas

O docente deseja alterar a data de uma prova prevista para amanhã, e remarca-la para o fim do mês, ele então acessa a disciplina, vai até a atividade pré-cadastrada, e então edita a data da mesma

* Editar/Consultar Disciplina

O responsável pela disciplina, deseja editar a descrição da disciplina, ele então entra no sistema, abre a disciplina em questão e edita a mesma conforme sua necessidade

* Postar Avisos

A secretária acadêmica deseja avisar os usuários do sistema que no mês seguinte haverá um evento, o responsável pelos avisos então, loga no sistema, vai até a aba de avisos, seleciona a opção “postar aviso”, configura o mesmo conforme sua necessidade e confirma a postagem

* Lançar Documentos

Um aluno deseja lançar no sistema, os comprovantes das doses que tomou da vacina de covid-19, o mesmo então, acessa a aba lançar documento, seleciona a opção, comprovante de vacinação dentre as opções desejadas, fornece as informações necessárias e confirma o lançamento

* Trocar Senha

Um usuário deseja alterar sua senha de acesso, o mesmo então, acessa o sistema, vai até a aba trocar senha, fornece a senha atual e a senha que deseja utilizar a partir de agora, após isso, ele confirma a alteração

* Trocar Email

Um aluno deseja editar o email que foi cadastrado no sistema, o mesmo acessa a aba trocar email, edita o email atualmente cadastrado e confirma o email através do email de confirmação enviado pelo sistema.

* Solicitar Documentos

Um aluno de GTI deseja solicitar um comprovante de matricula no curso, o mesmo então, acessa o sistema, vai até a aba solicitar documentos, e seleciona a opção deseja, o sistema realiza o envio do documento ao email cadastrado e confirma a aprovação da solicitação

* Solicitar Revisão

Se aproximando o fim do semestre, um aluno deseja contestar suas notas em uma determinada disciplina, o mesmo então, acessa o sistema, e seleciona a opção solicitar revisão, após o envio da solicitação e o termino do processo interno, o aluno recebe, no email cadastrado um retorno a respeito de sua solicitação

## 1.5 Ferramentas

As ferramentas utilizadas nessa etapa foram:

1. LucidChart

# INTERFACE

## Proposta de Interface

**Tela Inicial (Dashboard)**

A tela inicial do aplicativo deve apresentar um dashboard limpo e intuitivo que fornece um resumo rápido das informações mais importantes. Esta tela pode incluir:

* Foto e Nome do Aluno: Na parte superior, para identificação rápida.
* Resumo do Curso: Com informações como nome do curso, semestre atual, e status geral.
* Notificações Recentes: Uma seção com as notificações mais recentes, como prazos de matrícula, novos avisos, e eventos importantes.
* Acesso Rápido: Ícones para funcionalidades principais, como "Matrícula", "Histórico", "Notas", e "Calendário de Provas".

**Menu Lateral (Navigation Drawer)**

Um menu lateral deve estar acessível através de um ícone de hambúrguer (três linhas horizontais) no canto superior esquerdo. Este menu deve incluir todas as opções presentes na interface web, organizadas de forma lógica:

* Matrícula
* Consultas (com submenus: Aviso, Histórico, Histórico Completo, Horário, Notas Parciais, Faltas Parciais)
* Agenda (Calendário de Provas)
* Segurança (Troca de Senha)
* Solicitações (Nova/Nota/Falta/Exame Geral, Solicitação de Documentos, Regime de Exercícios Domiciliares)
* Biblioteca
* Upload de Arquivos

**Matrícula**

Na seção de matrícula, o usuário pode visualizar e gerenciar suas disciplinas. Devem estar disponíveis informações sobre disciplinas já cursadas, disciplinas disponíveis para matrícula, e status das matrículas anteriores.

**Consultas**

Dentro de consultas, o aluno pode acessar:

* Histórico: Visualização detalhada do histórico acadêmico com notas e disciplinas cursadas.
* Notas Parciais: Exibir notas e faltas parciais para cada disciplina no semestre atual.
* Horário: Horário das aulas, apresentado de forma semanal com horários claramente definidos.
* Avisos: Lista de avisos importantes emitidos pela instituição.

**Agenda**

O calendário de provas deve ser fácil de navegar, com eventos coloridos indicando diferentes tipos de provas e prazos. O aluno pode adicionar lembretes pessoais e receber notificações.

**Segurança**

Opção para troca de senha, com um fluxo simples e seguro para redefinição de senha.

**Solicitações**

Área onde o aluno pode fazer diferentes tipos de solicitações, como pedir documentos ou solicitar uma revisão de nota. Cada solicitação deve ter um formulário específico, fácil de preencher e enviar.

**Upload de Arquivos**

Uma interface simples para o upload de arquivos necessários, como comprovantes de vacinação. A possibilidade de tirar fotos diretamente da câmera do dispositivo ou selecionar da galeria deve estar disponível.

**Design e Usabilidade**

* Design Responsivo: Garantir que a interface se ajuste bem a diferentes tamanhos de tela e resoluções.
* Navegação Intuitiva: Facilitar a navegação com menus organizados e uma hierarquia clara de informações.
* Feedback Rápido: O aplicativo deve fornecer feedback imediato ao usuário após ações como envio de formulários ou upload de arquivos.
* Segurança: Implementar autenticação segura e garantir que dados sensíveis estejam protegidos.

## Ferramentas

Desenvolver uma interface mobile eficaz requer o uso de diversas ferramentas especializadas ao longo de diferentes etapas do processo de design. As ferramentas mais recomendadas para este propósito incluem Figma, Balsamiq, Proto.IO, ProtoPie.IO e Adobe XD. Cada uma delas oferece funcionalidades únicas que são ideais para diferentes fases do desenvolvimento.

**Concepção e Ideação**

**Balsamiq**

* Uso: Ideal para a fase inicial do design, Balsamiq permite a criação rápida de wireframes de baixa fidelidade.
* Etapa: Brainstorming e Ideação.
* Benefícios: Facilita a exploração de diferentes layouts e conceitos de maneira rápida e visualmente simplificada, sem a distração de detalhes visuais complexos.

**Figma**

* Uso: Criação de wireframes de média a alta fidelidade.
* Etapa: Concepção e Refinamento Inicial.
* Benefícios: Suporta colaboração em tempo real, permitindo que múltiplos designers trabalhem juntos e recebam feedback instantaneamente.

**Design de Interface**

**Figma**

* Uso: Criação de designs detalhados e de alta fidelidade.
* Etapa: Design de Interface.
* Benefícios: Componentes reutilizáveis e bibliotecas de estilos garantem consistência visual em toda a interface. A capacidade de prototipagem integrada ajuda a visualizar a navegação e as interações básicas.

**Adobe XD**

* Uso: Alternativa ou complementar ao Figma, Adobe XD também pode ser usado para criar interfaces detalhadas.
* Etapa: Design de Interface e Protótipos Interativos.
* Benefícios: Integração perfeita com outros produtos Adobe, permitindo importação e manipulação fácil de assets de Photoshop e Illustrator. Oferece recursos robustos de prototipagem.

**Prototipagem Interativa**

**Proto.IO**

* Uso: Utilizado para criar protótipos altamente interativos que simulam a experiência do usuário final.
* Etapa: Prototipagem Interativa.
* Benefícios: Permite a criação de protótipos realistas sem necessidade de codificação, facilitando testes de usabilidade e validação de design com usuários reais.

**ProtoPie.IO**

* Uso: Ideal para protótipos avançados que necessitam de interações complexas e animações detalhadas.
* Etapa: Prototipagem Avançada.
* Benefícios: Suporta a criação de interações condicionais e a utilização de sensores de dispositivos móveis, proporcionando um alto nível de realismo nos protótipos.

**Testes de Usabilidade**

**Figma**

* Uso: Utilizado para criar protótipos clicáveis para testes de usabilidade iniciais.
* Etapa: Testes de Usabilidade Básicos.
* Benefícios: A facilidade de compartilhamento e o feedback em tempo real ajudam na rápida iteração baseada em feedback de usuários.

Proto.IO e ProtoPie.IO

Uso: Ferramentas para testes de usabilidade mais avançados e interativos.

Etapa: Testes de Usabilidade Avançados.

Benefícios: Proto.IO e ProtoPie.IO permitem a criação de protótipos complexos que podem ser usados para testar interações detalhadas e comportamentos de usuário em condições próximas ao real.

## 2.3 Recomendações e Boas Práticas

O desenvolvimento de interfaces mobile eficazes exige a aplicação de heurísticas de usabilidade, como as propostas por Jakob Nielsen, para garantir uma experiência positiva ao usuário. Este artigo explora como essas diretrizes podem ser integradas ao processo de design de interfaces mobile utilizando ferramentas especializadas como Figma, Balsamiq, Proto.IO, ProtoPie.IO e Adobe XD. Cada ferramenta possui funcionalidades específicas que são ideais para diferentes fases do desenvolvimento, desde a concepção inicial até a prototipagem e teste.

**Concepção e Ideação**

Na fase inicial de concepção e ideação, Balsamiq se destaca como uma ferramenta ideal para a criação de wireframes de baixa fidelidade. Esta etapa é crucial para o brainstorming e a comunicação de ideias iniciais de design, permitindo a exploração de diferentes layouts de maneira rápida e eficaz. Balsamiq facilita a expressão de conceitos sem a necessidade de se preocupar com detalhes visuais complexos, o que é essencial para a iteração rápida e eficiente.

Simultaneamente, Figma pode ser utilizado para a criação de esboços de média a alta fidelidade. A capacidade de Figma de suportar colaboração em tempo real é particularmente valiosa nesta fase, permitindo que múltiplos designers trabalhem simultaneamente no mesmo projeto e recebam feedback instantâneo. Esta funcionalidade é crucial para assegurar que todos os membros da equipe estejam alinhados quanto aos objetivos e direções do design.

**Design de Interface**

Para o design detalhado da interface, Figma e Adobe XD são ferramentas altamente recomendadas. Ambas permitem a criação de interfaces de alta fidelidade com componentes reutilizáveis e consistentes. O uso de bibliotecas de estilos e componentes em Figma e Adobe XD garante que a interface mantenha uma consistência visual e funcional, essencial para a usabilidade.

A integração de Figma e Adobe XD com funcionalidades de prototipagem facilita a visualização de navegação e interações básicas, assegurando a aplicação de princípios como a visibilidade do status do sistema e a consistência de padrões. Essas ferramentas permitem que os designers criem e testem fluxos de trabalho, minimizando a possibilidade de erros do usuário e maximizando a eficiência da interface.

**Prototipagem Interativa**

Nas etapas de prototipagem interativa, Proto.IO e ProtoPie.IO são essenciais. Proto.IO permite a criação de protótipos altamente interativos sem a necessidade de codificação, o que é crucial para testar a usabilidade e validar designs antes do desenvolvimento final. Esta ferramenta é valiosa para criar protótipos realistas que podem ser usados para obter feedback de usuários em cenários próximos ao real.

Por outro lado, ProtoPie.IO é ideal para a criação de protótipos avançados que exigem interações complexas e animações detalhadas. A capacidade de ProtoPie.IO de suportar a utilização de sensores de dispositivos móveis, como giroscópios e acelerômetros, proporciona um alto nível de realismo nos protótipos, permitindo testes detalhados de usabilidade em condições realistas.

**Testes de Usabilidade**

Durante a fase de teste de usabilidade, é essencial que os protótipos permitam a obtenção de feedback fácil e rápido. Figma pode ser utilizado para criar protótipos clicáveis que permitem testes de usabilidade básicos, facilitando ajustes rápidos com base no feedback dos usuários. Para testes de usabilidade mais avançados, Proto.IO e ProtoPie.IO são adequados, pois permitem a criação de protótipos que simulam interações detalhadas e comportamentos de usuários em cenários complexos.

**Aplicação das Heurísticas de Nielsen**

Aplicar as heurísticas de Nielsen, como visibilidade do status do sistema, compatibilidade com o mundo real, controle e liberdade do usuário, consistência e padrões, prevenção de erros, reconhecimento em vez de recordação, flexibilidade e eficiência de uso, design estético e minimalista, ajuda na recuperação de erros e documentação acessível, é fundamental para garantir que a interface mobile seja intuitiva, eficiente e agradável.

# GESTÃO DE CONFIGURAÇÃO

## 3.1 Proposta de organização de configuração

Seguindo as funções básicas da Gerência de Configuração de Software, a proposta de organização será estruturada da seguinte maneira:

* Identificação de Configuração (quais itens de software constituem em configuração)

1. Código fonte
2. Documentos do Sistema
3. Tabelas de Parâmetros
4. Estruturas de Arquivos

* Controle de Configuração (que passos no processo de alteração afetam uma configuração);

O conceito de configuração básica (baseline) é um dos fundamentos da função de Controle de Configuração. A geração de uma configuração básica é o momento no qual é acertado que um ou mais itens de configuração estão em conformidade com os requisitos do projeto e devem ser protegidos contra alterações não autorizadas. Após o estabelecimento de uma configuração básica, toda e qualquer alteração sobre os itens desta configuração deve ser formalmente controlada.

* Auditoria de Configuração (quais são as diferenças entre as versões);

A auditoria de configuração é processo de examinar um produto por um grupo independente do grupo que o produziu. Pode se dizer que auditoria é o preço que se paga pela qualidade, no sentido de que através da auditoria assegura-se ao

usuário que o produto será desenvolvido de acordo com os requisitos do projeto. A auditoria deve ser executada antes que uma configuração básica seja definida, para certificar que o produto esteja realmente de acordo com os requisitos, tanto de especificação quanto contratuais, e também para garantir que o produto esteja precisamente descrito em seus documentos técnicos. Para garantir a integridade do produto que está sendo desenvolvido, toda alteração também precisa ser auditada, isto é, a auditoria é utilizada para tornar viável ao gerenciamento o status do software ao longo do ciclo de vida. Além disso, auditoria revela se os requisitos iniciais estão mesmo sendo satisfeitos e se a passagem de uma configuração a outra não descaracteriza o produto. Para a realização do trabalho de auditoria, utiliza-se os

procedimentos de verificação e validação.

* Administração de Estados (quais modificações foram feitas e por quem).

A administração de estados é processo que permite obter informações sobre o estado das configurações-base. A obtenção de informações sobre o estado da configuração deve começar assim que os itens de configurações forem identificados. A administração de estado permite localizar as modificações ocorridas durante o processo de desenvolvimento do software, possibilitando identificar o autor da modificação proposta, tipo de modificação e o momento da modificação efetuada.

## 3.2 Criação de projeto e inclusão de documentos

O projeto contará com os seguintes documentos

* Suporte ao usuário final

Um guia de usuário, com notas de versão, sistemas de ajuda online, programas de treinamento ou procedimentos operacionais, isto é, qualquer detalhe que ajude os usuários a usar o produto.

* Suporte de marketing

Uma publicidade focada no produto, usada para promover a marca e a empresa, como fotos, vídeos explicativos, apresentações e materiais que levam a páginas de destino com informações técnicas.

* Suporte ao desenvolvimento

Especificações funcionais e técnicas, que servem como guias de desenvolvimento de software ou simplesmente procedimentos e ferramentas para ajudar desenvolvedores a fazer seu trabalho.

* Suporte à organização

Informações sobre a empresa, sua estrutura, seus procedimentos, fluxos de trabalho, as políticas e tudo o mais que os profissionais da sua equipe precisam saber para realizar seus trabalhos.

## 3.3 Ferramentas

As ferramentas utilizadas nessa etapa foram:

1. Microsoft Word
2. Microsoft Sharepoint
3. Microsoft Teams
4. Adobe Photoshop

# MÉTRICAS

## 4.1 Pontos de função

Dentre os pontos de função não ajustados, serão considerados os seguintes para o desenvolvimento do sistema:

* Comunicação de Dados
* Processamento Distribuído
* Performance
* Configuração Intensamente Utilizada
* Volume de Transações
* Entrada de Dados On-Line
* Eficiência do Usuário Final
* Atualização On-Line
* Processamento Complexo
* Reusabilidade
* Facilidade de Instalação
* Facilidade de Operação
* Múltiplos Locais
* Facilidade de Mudança

## 4.2 Métricas para desenvolvimento

Dentre as diversos tipos de métricas e métricas existentes, serão utilizadas as seguintes no desenvolvimento

* Métricas de produtividade do desenvolvedor

As métricas de produtividade permitem aos gerentes de desenvolvimento executarem melhor os projetos. uma combinação destas métricas de software é utilizada para medir o quão avançado está um projeto, os níveis de produtividade do desenvolvedor, o tempo de desenvolvimento adicional necessário, e muito mais.

1. Tempo de entrega (lead time)

O tempo de entrega é o tempo que algo leva do princípio ao fim. No desenvolvimento de software, por exemplo, o tempo de execução de um projeto começa com a proposta e termina com a entrega.

1. Quantidade de código

As equipes de desenvolvimento podem usar esta métrica de software, também conhecida como KLOC (Milhares de linhas de código), para determinar o tamanho de um aplicativo. Se este KPI for elevado, poderia indicar que os desenvolvedores foram eficientes em seus esforços de programação. Contudo, essa métrica não é útil quando uma equipe de desenvolvimento tenta comparar dois projetos elaborados com diferentes linguagens de programação. Além disso, vale considerar que mais quantidade de código nem sempre torna o código eficiente ou eficaz, o que pode significar mais trabalho de refatoração depois.

1. Trabalho em progresso (WIP)

Em um contexto de engenharia de software, o WIP é um trabalho de desenvolvimento em que a equipe começou a trabalhar e já não se encontra em backlog. Um computador pode expressar o WIP num gráfico de burndown. Uma ferramenta comum para sprints ágeis e Scrum, esses gráficos mostram o quanto de trabalho a equipe já realizou e quanto ainda resta por fazer.

1. Velocidade ágil

Para calcular a velocidade, uma equipe de desenvolvimento ágil de software analisa os sprints anteriores e conta o número de histórias de usuários ou pontos de histórias completados ao longo do tempo. A velocidade ágil é uma estimativa de quão produtiva a equipe será num único sprint.

1. Taxa de sucesso da meta Sprint

Essa métrica de software calcula a porcentagem de itens que a equipe de desenvolvimento completou no sprint backlog. Pode ocorrer que uma equipe não complete 100% do trabalho durante um determinado sprint. No entanto, o progresso da equipe poderia ainda cumprir a sua Definição de pronto (DoD): o limiar que um projeto deve cumprir para que uma organização o considere concluído. Se a iteração de desenvolvimento cumprir com definição de pronto, já é um sucesso.

1. Número de versões de software

As equipes de desenvolvimento ágil e os DevOps priorizam os lançamentos de software contínuos e frequentes. Com esse KPI, as equipes podem monitorar com que frequência lançam software, seja mensal, semanal, diária, por hora, ou qualquer outro período de tempo, e se esse ritmo proporciona valor comercial suficiente.

* Métricas de desempenho de software

O desempenho do software refere-se a medidas quantitativas do comportamento de um sistema de software. As métricas de desempenho medem atributos não funcionais, ou seja, qual é a eficiência de um aplicativo, não o que ele faz. Aspectos do desempenho de software. Os testes de desempenho podem avaliar as seguintes características de um aplicativo:

1. Escalabilidade
2. Estabilidade
3. Capacidade de resposta
4. Velocidade
5. Disponibilidade

Outras expressões importantes de métricas de desempenho de software incluem o seguinte:

1. Desempenho (throughput)

O desempenho é a quantidade de unidades de dados que um sistema processa num determinado período de tempo.

1. Tempo de resposta

O tempo de resposta mede o tempo que um sistema leva para responder uma consulta ou demanda.

1. Confiabilidade, disponibilidade e capacidade de assistência (RAS)

RAS refere-se à capacidade do software de cumprir consistentemente com suas especificações; quanto tempo funciona em relação à quantidade esperada; e o grau de facilidade com que pode ser reparado ou mantido.

* Métricas baseadas em defeitos

As equipes de desenvolvimento devem entender as falhas dos aplicativos para poder aperfeiçoá-los. Estas métricas de desenvolvimento de software avaliam defeitos e vulnerabilidades.

1. Densidade de defeitos

Em termos de código, os desenvolvedores podem tabular o número de defeitos por KLOC para avaliar a frequência dos defeitos.

1. Cobertura de código

Esta é a proporção de código fonte coberta por testes automatizados. A métrica do software permite aos testadores identificar quais as áreas do código que ainda devem ser testadas corretamente.

1. Taxa de detecção de defeitos

Essa métrica é uma relação entre o número de defeitos encontrados antes do lançamento do software, em comparação com o número encontrado após o lançamento. Para calcular a porcentagem, deve-se anotar o número de defeitos encontrados antes do lançamento (x) e o número encontrado pelos usuários após o lançamento (y), e depois calcular x/(x + y). Um alto percentual é preferível, pois significa que uma maior proporção de defeitos foi encontrada antes que os clientes utilizassem o software.

1. Dívida técnica

A dívida técnica é uma metáfora que reflete o esforço a longo prazo, bem como os custos temporais e financeiros, dos desenvolvedores que não abordam um problema de desenvolvimento quando este surge pela primeira vez.

1. Vulnerabilidades de segurança

As análises de vulnerabilidade identificam as falhas de segurança em um aplicativo. Quanto menor for o número de vulnerabilidades encontradas, mais seguro é o software.

1. Incidentes de segurança reais

Esse KPI conta o número de vezes que um hacker explora uma vulnerabilidade no software. Acompanha a frequência com que essas infrações ocorrem, a gravidade do ataque (por exemplo, que dados foram roubados), e o tempo de duração do incidente.

As organizações de TI utilizam várias médias para calcular a ocorrência de falhas ou defeitos de software:

* Tempo médio de detecção: O tempo médio de detecção é uma média que indica quanto tempo leva uma equipe para detectar um problema ou erro.
* Tempo médio entre falhas. Essa métrica se remete ao cálculo do tempo transcorrido entre uma falha e a próxima que irá ocorrer.
* Tempo médio para reparo: O tempo médio para reparo é a média que representa a rapidez com que uma equipe soluciona as falhas.
* Métricas de usabilidade e UX

Os usuários experimentam e interagem com o software de diferentes maneiras. Assim como é difícil classificar as emoções das pessoas, também é um desafio avaliar como elas reagem ao software. Embora nenhuma métrica de software possa informar a totalidade da experiência do usuário, há algumas que são úteis:

1. As métricas de UX costumam ser qualitativas e podem incluir respostas emocionais ou corporais dos usuários, tais como o quanto confiam no software e como os seus olhos se movem através de uma interface de usuário.
2. Métricas de usabilidade. A usabilidade mede quão bem o software permite aos clientes alcançarem seus objetivos. A usabilidade inclui os seguintes componentes:

* Facilidade de descoberta
* Eficiência
* Memorabilidade
* Facilidade de aprendizagem
* Satisfação
* Acessibilidade, particularmente a acessibilidade digital
* Net Promoter Score (NPS). Essa métrica de software reflete a vontade dos clientes de recomendar um aplicativo a outros. O NPS é apresentado em uma escala de 0 a 10. Os clientes com uma pontuação de 0 a 6 são chamados Detratores; com pontuações de 7 e 8 são Neutros; e de 9 e 10 são Promotores.

## 4.3 Ferramentas

As ferramentas utilizadas nessa etapa do desenvolvimento foram:

1. Sizify
2. Smartsheet

# SCRUM

A metodologia SCRUM é uma abordagem ágil para o desenvolvimento de software que promove a entrega incremental de funcionalidades através de iterações chamadas sprints. A definição do backlog do produto é uma das etapas mais importantes no SCRUM, pois ela detalha todas as funcionalidades e tarefas necessárias para completar o projeto. A seguir, é apresentada uma definição do backlog do sistema para um aplicativo mobile baseado nas heurísticas de Jakob Nielsen e utilizando as ferramentas mencionadas (Figma, Balsamiq, Proto.IO, ProtoPie.IO e Adobe XD).

## Backlog do Sistema

O backlog do sistema é a lista prioritária de todas as funcionalidades e tarefas necessárias para o desenvolvimento do aplicativo SIGA. Ele serve como um guia detalhado para a equipe de desenvolvimento, ajudando a organizar o trabalho de maneira eficiente. No caso do aplicativo SIGA, o backlog pode incluir itens como a criação de interfaces de login e registro, desenvolvimento de funcionalidades para visualização de histórico acadêmico, calendário de provas, upload de documentos, sistema de notificações e outras funcionalidades essenciais. Cada item do backlog é detalhado com critérios de aceitação claros para garantir que todos entendam o que é necessário.

## Sprints

**Sprint 1:** Tela de Login e Registro

* Desenvolver formulários de login e registro.
* Implementar autenticação de usuário.
* Validar inputs e mensagens de erro.

**Sprint 2:** Painel Principal

* Desenvolver o layout do painel principal.
* Implementar navegação básica entre telas.
* Exibir informações principais (notificações, status do usuário).

**Sprint 3:** Calendário de Provas

* Desenvolver a interface do calendário.
* Implementar funcionalidade de visualização de provas e eventos.
* Adicionar opções de filtro e pesquisa.

**Sprint 4:** Histórico Acadêmico

* Desenvolver a tela de histórico acadêmico.
* Implementar a visualização de notas e cursos completados.
* Adicionar a funcionalidade de download de históricos em PDF.

**Sprint 5:** Upload de Arquivos

* Desenvolver a funcionalidade de upload de documentos (comprovantes de vacinação, etc.).
* Implementar validações e confirmações de upload.

**Sprint 6:** Sistema de Notificações

* Desenvolver a funcionalidade de envio e recebimento de notificações.
* Implementar configurações de notificação personalizáveis.

## Simulação 1° Sprint

**Sprint 1: Tela de Login e Registro**

Objetivo: Desenvolver as telas de login e registro e implementar a autenticação do usuário.

Tarefas e Atividades:

* Criação de Wireframes
* Utilizar Balsamiq para desenvolver wireframes de baixa fidelidade das telas de login e registro.
* Design de Interface
* Utilizar Figma e Adobe XD para criar protótipos de alta fidelidade.
* Definir layout e estética visual.
* Desenvolvimento do Backend
* Configurar o servidor backend usando Node.js e Express.
* Implementar endpoints de API para login, registro e recuperação de senha.
* Desenvolvimento do Frontend
* Desenvolver componentes de formulário usando React Native.
* Implementar chamadas de API para autenticação.
* Validação de Inputs
* Adicionar validações de campo e feedback em tempo real.
* Testes e Debugging
* Criar testes unitários e de integração.
* Testar endpoints de API com Postman.
* Revisão e Refinamento
* Realizar revisões das funcionalidades desenvolvidas.
* Ajustar design e funcionalidade com base no feedback.
* Preparação para a Próxima Sprint
* Documentar o trabalho realizado.
* Planejar as atividades para a Sprint 2.

## Burndown Chart

Um “burndown chart” é uma ferramenta visual que mostra o progresso do trabalho ao longo do tempo. Ele é usado para monitorar o progresso da sprint, mostrando a quantidade de trabalho restante versus o tempo disponível.

Passos para gerar um burndown chart:

* **Eixo X (Horizontal):** Representa os dias da sprint.
* **Eixo Y (Vertical):** Representa o trabalho restante (em horas ou story points).
* **Linha Ideal:** Uma linha diagonal que vai do trabalho total no início da sprint até zero no final da sprint.
* **Linha Real:** Atualizada diariamente, mostra o progresso real do trabalho.

A equipe atualiza o burndown chart diariamente durante as reuniões de stand-up, registrando o trabalho completado e o trabalho restante. Isso ajuda a identificar se o time está no caminho certo para completar todas as tarefas da sprint e permite ajustes rápidos se necessário.

## 5.5 Ferramentas

Para gerenciar o projeto SCRUM do aplicativo SIGA, várias ferramentas podem ser utilizadas, cada uma oferecendo funcionalidades específicas que facilitam o gerenciamento ágil:

**Jira Agile:**

* Ferramenta robusta de gerenciamento de projetos que oferece suporte completo para SCRUM.
* Permite a criação e gestão de backlogs, sprints e burndown charts.
* Funcionalidades avançadas de relatórios e integração com outras ferramentas de desenvolvimento.

**Trello:**

* Ferramenta intuitiva baseada em kanban boards, adequada para equipes pequenas e médias.
* Permite a criação de cartões para tarefas, listas para sprints e etiquetas para priorização.
* Integrável com outras ferramentas de produtividade e comunicação.

**PangoSCRUM:**

* Ferramenta específica para gerenciamento SCRUM, focada em simplicidade e eficiência.
* Facilita a criação de backlogs, sprints e relatórios de progresso.
* Ideal para equipes que desejam uma solução focada exclusivamente em SCRUM.

**Monday:**

* Plataforma versátil de gerenciamento de projetos que suporta várias metodologias ágeis, incluindo SCRUM.
* Oferece funcionalidades visuais e personalizáveis para gerenciamento de tarefas e projetos.
* Facilita a colaboração em tempo real e a integração com outras ferramentas.

**GitHub Projects:**

* Ferramenta integrada ao GitHub, permitindo o gerenciamento de projetos diretamente junto ao código-fonte.
* Suporte para kanban boards, backlogs e sprints.
* Facilita a rastreabilidade e o controle de versões de software.

# TESTES

## Testes para o sistema

Dentre os diversos tipos de testes, serão utilizados os seguintes testes

1. Testes de unidade

Testes de unidade são feitos em um nível muito baixo, próximo ao código-fonte do aplicativo. Eles consistem em testar métodos e funções individuais de classes, componentes ou módulos usados pelo software. Testes de unidade, em geral, têm custo baixo para automatizar e podem ser executados com rapidez por um servidor de integração contínua.

1. Testes de integração

Testes de integração verificam se diferentes módulos ou serviços usados pelo seu aplicativo funcionam bem juntos. Por exemplo, pode ser testar a interação com o banco de dados ou garantir que os microsserviços funcionem juntos conforme o esperado. A execução desses tipos de testes tem um custo maior, uma vez que exigem que várias partes do aplicativo estejam ativas e em execução.

1. Testes funcionais

Os testes funcionais têm como foco os requisitos de negócios de uma aplicação. Eles só verificam a saída de uma ação e não verificam os estados intermediários do sistema ao executar essa ação.

Às vezes há uma confusão entre testes de integração e testes funcionais, uma vez que ambos exigem vários componentes para interagirem entre si. A diferença é que um teste de integração pode simplesmente verificar que você pode consultar o banco de dados, enquanto um teste funcional esperaria obter um valor específico do banco de dados conforme definido pelos requisitos do produto.

1. Testes de ponta a ponta

Teste de ponta a ponta replica o comportamento de um usuário com o software em um ambiente de aplicativo completo. Ele verifica se vários fluxos de usuário funcionam como o esperado e podem ser tão simples quanto carregar uma página da web ou fazer login ou cenários muito mais complexos verificando notificações por e-mail, pagamentos on-line etc.

Testes de ponta a ponta são muito úteis, mas têm um alto custo e podem ser difíceis de atualizar quando automatizados. Recomendamos ter alguns testes de ponta a ponta essenciais e contar mais com tipos de testes de nível inferior (testes de unidade e de integração) para poder identificar rapidamente alterações que causam falha.

1. Testes de aceitação

Os testes de aceitação são testes formais executados para verificar se um sistema atende aos requisitos de negócios. Eles exigem que todo o aplicativo esteja ativo e em execução e foca em replicar os comportamentos do usuário. Porém, também pode ir mais além e medir o desempenho do sistema e rejeitar alterações se determinadas metas não forem cumpridas.

1. Testes de desempenho

Os testes de desempenho avaliam o desempenho de um sistema sob uma carga de trabalho específica. Esses testes ajudam a medir a confiabilidade, a velocidade, a escalabilidade e a capacidade de resposta de um aplicativo. Por exemplo, o teste de desempenho pode observar tempos de resposta ao executar um grande número de solicitações, ou ver como o sistema se comporta com quantidade significativa de dados. Ele pode determinar se um aplicativo atende aos requisitos de desempenho, localizar gargalos, medir a estabilidade durante picos de tráfego e muito mais.

1. Teste de fumaça (smoke test)

Testes de sanidade são testes básicos que verificam a funcionalidade básica do aplicativo. Eles são feitos para terem execução rápida e sua meta é garantir que os principais recursos do seu sistema estejam funcionando conforme o esperado.

Os testes de sanidade podem ser úteis logo após um novo build ser feito para decidir se você pode ou não executar testes mais caros, ou logo após uma implementação para garantir que o aplicativo esteja sendo executado como deveria no ambiente recém-implementado.

## 6.2 Ferramentas

Nessa etapa do desenvolvimento, seão utilizadas as seguintes ferramentas:

1. Atlassian
2. Jira