



FACULDADE DE TECNOLOGIA DO IPIRANGA
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GABRIEL OLIVEIRA PAULON
THIBOR DE MATTOS MARTIN

G DEV FLOW – PLATAFORMA DE GERENCIAMENTO E
DESENVOLVIMENTO DE PROJETOS

SÃO PAULO
2026

FACULDADE DE TECNOLOGIA DO IPIRANGA
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GABRIEL OLIVEIRA PAULON
THIBOR DE MATTOS MARTIN

**G DEV FLOW – PLATAFORMA DE GERENCIAMENTO E
DESENVOLVIMENTO DE PROJETOS**

Trabalho de Conclusão de Curso
apresentado à Faculdade de Tecnologia
do Ipiranga, como requisito parcial para a
obtenção do grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas.

Docente: Prof(a). Dra. Ana Claudia Melo
Tiesse Gomes de Oliveira.

SÃO PAULO
2026

GABRIEL OLIVEIRA PAULON
THIBOR DE MATTOS MARTIN

**G DEV FLOW – PLATAFORMA DE GERENCIAMENTO E
DESENVOLVIMENTO DE PROJETOS**

Trabalho de Conclusão de Curso
apresentado à Faculdade de Tecnologia
do Ipiranga, como requisito parcial para a
obtenção do grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas.

Data de aprovação:
Banca examinadora:

Prof(a).

Prof(a).

Orientador:

SÃO PAULO

2026

RESUMO

O gerenciamento de projetos de software entre desenvolvedores freelancers e clientes apresenta desafios relacionados à organização das atividades, à comunicação entre as partes e ao acompanhamento do progresso do desenvolvimento. A ausência de ferramentas adequadas pode dificultar a visualização das etapas do projeto e o alinhamento entre as expectativas do cliente e as entregas realizadas. Diante desse cenário, este trabalho apresenta o desenvolvimento do G Dev Flow, uma plataforma digital voltada ao gerenciamento de projetos entre freelancers e clientes. A aplicação permite ao desenvolvedor organizar o desenvolvimento por meio de criação de projetos, sprints e tarefas, além de registrar atualizações e realizar a comunicação com o cliente. Ao cliente, a plataforma possibilita o acompanhamento do projeto, a interação por meio do chat com desenvolvedor e a validação final do projeto. O sistema foi desenvolvido utilizando uma arquitetura composta por aplicação móvel e backend, empregando as tecnologias React Native, Spring Boot e PostgreSQL, com o objetivo de garantir organização, persistência de dados e comunicação eficiente entre os usuários. Espera-se que a solução contribua para aumentar a transparência no desenvolvimento de projetos freelancers, facilitar o acompanhamento das atividades e promover uma comunicação mais eficiente entre desenvolvedor e cliente.

Palavras-chave: Desenvolvimento de Software, Freelancers, Gerenciamento de Projetos, Plataforma Digital, Sprints.

ABSTRACT

Software project management between freelance developers and clients presents challenges related to task organization, communication between stakeholders, and monitoring development progress. The lack of appropriate tools can make it difficult to visualize project stages and align client expectations with delivered results. In this context, this study presents the development of G Dev Flow, a digital platform designed to support project management between freelancers and clients. The application enables developers to organize their work through the creation of projects, sprints, and tasks, as well as to register updates and communicate with clients. For clients, the platform provides project monitoring, interaction through a chat feature with the developer, and final project validation. The system was developed using a mobile application and backend architecture, employing React Native, Spring Boot, and PostgreSQL to ensure organization, data persistence, and efficient communication between users. It is expected that the proposed solution will increase transparency in freelance software projects, facilitate activity tracking, and promote more effective communication between developers and clients.

Keywords: Freelancers, Project Management, Software Development, Digital Platform, Sprints.

LISTA DE FIGURA

Figura 1 – Modelo Entidade Relacionamento (MER).....	19
Figura 2 – Diagrama de Casos de Uso	20
Figura 3 – Diagrama de Classes.....	27
Figura 4 – Diagrama de Sequência - Autenticar Usuário	28
Figura 5 – Diagrama de Sequência - Gerenciar Projeto.....	29
Figura 6 – Diagrama de Sequência - Gerenciar Sprints.....	29
Figura 7 – Diagrama de Sequência - Gerenciar Tarefas.....	30
Figura 8 – Diagrama de Sequência - Visualizar Progresso.....	30
Figura 9 – Diagrama de Sequência - Registrar Tarefa Concluída.....	31
Figura 10 – Diagrama de Arquitetura.....	32
Figura 11 – Ecossistema	34
Figura 12 – Tela - Cadastro.....	42
Figura 13 – Tela - Autenticar usuário.....	43
Figura 14 – Tela - Criar projeto.....	44
Figura 15 – Tela -Gerenciar projeto.....	45
Figura 16 – Tela - Criar sprint.....;	46
Figura 17 – Tela - Gerenciar sprint.....	47
Figura 18 – Tela - Criar tarefa.....	48
Figura 19 – Tela - Registrar tarefa concluída.....	49
Figura 20 – Tela - Visualizar progresso do projeto.....	50
Figura 21 – Tela - Chat do projeto.....	51

LISTA DE TABELAS

Tabela 1 – Especificação do Caso de Uso UC01	21
Tabela 2 – Especificação do Caso de Uso UC02	22
Tabela 3 – Especificação do Caso de Uso UC03	23
Tabela 4 – Especificação do Caso de Uso UC04.....	24
Tabela 5 – Especificação do Caso de Uso UC05	25
Tabela 6 – Especificação do Caso de Uso UC06	26
Tabela 7 - Roteiro de Testes UC1 - Autenticar Usuário	36
Tabela 8 - Roteiro de Testes UC2 - Gerenciar Projetos	37
Tabela 9 - Roteiro de Testes UC3 - Gerenciar Sprints	38
Tabela 10 - Roteiro de Testes UC4 - Gerenciar Tarefas	39
Tabela 11 - Roteiro de Testes UC5 - Visualizar Progresso do Projeto.....	40
Tabela 12 - Roteiro de Testes UC6 - Registrar Tarefa Concluída.....	41

SUMÁRIO

RESUMO.....	3
ABSTRACT.....	4
1. INTRODUÇÃO.....	9
1.1. Objetivo Geral.....	10
1.2. Objetivos Específicos.....	10
1.3. Justificativa.....	11
1.4. Organização do Documento.....	12
2. PROJETO TÉCNICO.....	13
2.1. Regras de Negócio (RN).....	13
2.2. Requisitos Funcionais (RF).....	14
2.3. Requisitos Não Funcionais (RNF).....	16
2.4. Modelagem do Banco de Dados.....	18
2.5. Diagrama de Casos de Uso.....	19
2.5.1. Especificação de Casos de Uso.....	21
2.6. Diagrama de Classes.....	27
2.7. Diagrama de Sequência.....	28
2.8. Diagrama de Arquitetura.....	32
3 IMPLEMENTAÇÃO.....	33
3.1 Descrição da Aplicação.....	33
3.2 Implantação.....	35
3.3 Roteiro de Testes.....	35
3.4 Telas.....	42
4 RESULTADOS.....	53
4.1 Descrição dos resultados.....	53
4.2 Perspectivas Futuras.....	54
REFERÊNCIAS.....	56

1. INTRODUÇÃO

O desenvolvimento de software envolve diversas etapas e exige organização, comunicação eficiente e acompanhamento contínuo das atividades. No contexto de projetos realizados por desenvolvedores freelancers, esses desafios podem ser ainda mais evidentes, especialmente na relação direta com o cliente, que muitas vezes não possui visibilidade clara sobre o andamento do projeto.

De acordo com o Project Management Institute (2017), um projeto é caracterizado como um esforço temporário com o objetivo de criar um produto, serviço ou resultado único, sendo necessária a aplicação de técnicas de planejamento, execução e controle para garantir seu sucesso. Nesse sentido, a gestão de projetos torna-se essencial para assegurar o cumprimento de prazos, qualidade das entregas e alinhamento entre as partes envolvidas.

Embora existam ferramentas voltadas para o gerenciamento de projetos, muitas delas são direcionadas a grandes equipes ou apresentam complexidade desnecessária para o contexto de trabalho freelancer. Como consequência, problemas como falta de transparência, dificuldades na comunicação entre cliente e desenvolvedor e ausência de acompanhamento estruturado do progresso podem impactar negativamente a qualidade das entregas e a satisfação do cliente. Segundo Sommerville (2011), a qualidade do processo de desenvolvimento está diretamente relacionada à qualidade do produto final, evidenciando a importância de práticas organizadas e bem definidas.

Diante desse cenário, este trabalho propõe o desenvolvimento do G Dev Flow, uma plataforma voltada para o gerenciamento de projetos entre desenvolvedores freelancers e clientes. O sistema tem como objetivo proporcionar maior organização e transparência no desenvolvimento de projetos, permitindo que o freelancer estruture o trabalho por meio de projetos, sprints, tarefas e subtarefas, além de registrar atualizações.

A proposta do sistema também se relaciona com os princípios das metodologias ágeis, que priorizam a colaboração entre cliente e desenvolvedor, bem como o acompanhamento contínuo das atividades. Conforme apresentado

por Schwaber e Sutherland (2020), o uso de ciclos iterativos, como sprints, permite maior controle e adaptação ao longo do desenvolvimento, favorecendo a entrega incremental de valor.

A plataforma possibilita que o cliente acompanhe o progresso do projeto em tempo real, interaja por meio de comentários e valide as entregas realizadas, promovendo uma comunicação mais clara e eficiente entre as partes envolvidas.

Este trabalho apresenta o desenvolvimento da aplicação, abordando o problema identificado, a proposta de solução, os requisitos do sistema, a modelagem da aplicação e as tecnologias utilizadas para sua implementação.

1.1. Objetivo Geral

Neste projeto o objetivo é desenvolver uma plataforma digital denominada G Dev Flow, voltada para o gerenciamento de projetos de software entre desenvolvedores freelancers e clientes, com o objetivo de promover maior organização das atividades, transparência no acompanhamento do progresso e melhoria na comunicação entre as partes envolvidas.

1.2. Objetivos Específicos

Neste projeto foram definidos alguns objetivos específicos que orientam o desenvolvimento do sistema, entre os principais objetivos estão o desenvolvimento de um sistema que permita o cadastro e autenticação de usuários, possibilitando o acesso seguro à plataforma. Além disso, busca-se implementar funcionalidades que permitam a criação e o gerenciamento de projetos dentro do sistema.

Outro objetivo importante é permitir a criação e organização de tarefas relacionadas aos projetos, possibilitando que essas atividades sejam atribuídas aos membros da equipe responsáveis por sua execução. Dessa forma, torna-se possível acompanhar o andamento das tarefas e visualizar o progresso das atividades do projeto.

Por fim, pretende-se desenvolver uma interface que facilite a visualização das informações e contribua para uma melhor organização do trabalho em equipe, permitindo que os usuários tenham uma visão clara das atividades que precisam ser realizadas.

1.3. Justificativa

O desenvolvimento de software envolve diversas etapas e exige organização, comunicação eficiente e acompanhamento contínuo das atividades. Nesse contexto, a adoção de práticas modernas de desenvolvimento torna-se essencial para garantir a qualidade das entregas e a eficiência do processo. De acordo com o guia oficial do Scrum, a organização do trabalho em ciclos iterativos permite maior controle, adaptação e transparência ao longo do desenvolvimento (SCHWABER; SUTHERLAND, 2020).

Além disso, o uso de arquiteturas modernas baseadas em APIs REST tornou-se padrão no desenvolvimento de aplicações, pois possibilita a separação entre cliente e servidor, aumentando a escalabilidade e flexibilidade dos sistemas. Conforme definido por Fielding (2000), o estilo arquitetural REST favorece a simplicidade, a interoperabilidade e a independência entre as camadas da aplicação.

Complementarmente, boas práticas de engenharia de software recomendam o uso de arquiteturas desacopladas e comunicação baseada em serviços, facilitando a manutenção e evolução dos sistemas ao longo do tempo (IBM, 2023).

Apesar da existência de diversas ferramentas de apoio à gestão de projetos, muitas delas são voltadas a ambientes corporativos ou apresentam complexidade elevada para contextos mais simples, como o trabalho freelancer. Nesse cenário, torna-se relevante a aplicação de conceitos consolidados de metodologias ágeis, arquitetura de software e organização de tarefas, adaptados a diferentes realidades.

Dessa forma, este trabalho se justifica pela necessidade de aplicar, na prática, conceitos amplamente difundidos na área de desenvolvimento de software, utilizando abordagens como Scrum, organização por tarefas e acompanhamento contínuo, com o objetivo de melhorar a transparência, a comunicação e o controle das atividades ao longo do ciclo de desenvolvimento.

1.4. Organização do Documento

Este trabalho está organizado em capítulos que apresentam, de forma progressiva, a proposta, a estrutura técnica, o desenvolvimento e os resultados obtidos com a implementação da plataforma G Dev Flow.

No primeiro capítulo, são apresentados a introdução do trabalho, o contexto do problema identificado, o objetivo geral, os objetivos específicos e a justificativa para o desenvolvimento da solução proposta.

No segundo capítulo, é apresentado o projeto técnico do sistema, contemplando as regras de negócio, os requisitos funcionais e não funcionais, a modelagem do banco de dados, o diagrama de casos de uso, as especificações dos casos de uso, o diagrama de classes, os diagramas de sequência e o diagrama de arquitetura da aplicação.

No terceiro capítulo, é descrita a implementação da plataforma, abordando a arquitetura utilizada, as tecnologias empregadas no desenvolvimento, a estrutura da aplicação, os roteiros de testes elaborados para validar as principais funcionalidades e as telas do sistema.

No quarto capítulo, são apresentados os resultados obtidos a partir do desenvolvimento da aplicação, destacando as funcionalidades implementadas, os benefícios observados em relação à organização e ao acompanhamento de projetos, além das perspectivas futuras para evolução da plataforma.

Por fim, são apresentadas as referências utilizadas como base para a elaboração do trabalho.

2. PROJETO TÉCNICO

Esta seção apresenta os aspectos técnicos do trabalho, descrevendo as regras de negócio, requisitos funcionais e não funcionais, além da modelagem do banco de dados e dos diagramas que representam o sistema.

2.1. Regras de Negócio (RN)

[RN001] – Vínculo de Projeto

Descrição:

Cada projeto deve estar vinculado a um único desenvolvedor freelancer responsável por sua criação e gerenciamento.

[RN002] – Escopo do Papel do Cliente

Descrição:

O cliente possui acesso ao acompanhamento do progresso do projeto, podendo visualizar sprints e tarefas, além de validar a entrega final.

[RN003] – Escopo do Papel do Freelancer

Descrição:

Somente o freelancer pode criar, editar e gerenciar sprints, tarefas e demais elementos do projeto.

[RN004] – Associação Obrigatória de Tarefas

Descrição:

Toda tarefa deve estar obrigatoriamente associada a uma sprint e a um projeto.

[RN005] – Controle de Status da Tarefa

Descrição:

Cada tarefa deve possuir um status que represente seu estágio no fluxo de trabalho.

[RN006] – Atualização do Progresso

Descrição:

A conclusão das tarefas deve impactar automaticamente o progresso da sprint e do projeto.

[RN007] – Comunicação via Chat

Descrição:

A plataforma deve permitir a comunicação direta entre cliente e freelancer por meio de um chat, possibilitando a troca de mensagens durante o desenvolvimento do projeto.

[RN008] – Regra de Encerramento do Projeto

Descrição:

O projeto só poderá ser encerrado quando não houver tarefas pendentes nas sprints e após validação final pelo cliente.

2.2. Requisitos Funcionais (RF)

[RF001] – Cadastro de Usuário

Prioridade: Essencial

Descrição: O sistema deve permitir que novos usuários realizem seu cadastro no sistema, informando dados como nome, e-mail e senha. Após o cadastro, o usuário poderá acessar a plataforma utilizando suas credenciais.

[RF002] – Autenticação de Usuário

Prioridade: Essencial

Descrição: O sistema deve permitir que usuários registrados acessem a plataforma utilizando credenciais individuais (login e senha), garantindo autenticação segura e identificação do usuário.

[RF003] – Associação de Papel ao Usuário

Prioridade: Essencial

Descrição: O sistema deve associar cada usuário a um papel específico (Freelancer ou Cliente), definindo automaticamente suas permissões dentro da plataforma.

[RF004] – Gerenciamento de Projetos pelo Freelancer

Prioridade: Essencial

Descrição: O sistema deve permitir que o usuário com papel de Freelancer crie, edite e gerencie seus próprios projetos na plataforma.

[RF005] – Criação de Sprints

Prioridade: Essencial

Descrição: O sistema deve permitir que o Freelancer crie sprints dentro de um projeto.

[RF006] – Criação e Gerenciamento de Tarefas

Prioridade: Essencial

Descrição: O sistema deve permitir que o Freelancer crie, edite e organize tarefas associadas às sprints.

[RF007] – Atualização de Status das Tarefas

Prioridade: Essencial

Descrição: O sistema deve permitir que o Freelancer atualize o status das tarefas conforme o andamento do desenvolvimento.

[RF008] – Visualização do Progresso do Projeto

Prioridade: Importante

Descrição: O sistema deve permitir que o Cliente visualize o progresso do projeto, incluindo informações sobre sprints e tarefas.

[RF009] – Registro de Tarefas Concluídas

Prioridade: Importante

Descrição: O sistema deve permitir que o Freelancer registre tarefas concluídas e marcos de entrega do projeto.

[RF010] – Sistema de Chat

Prioridade: Importante

Descrição: O sistema deve permitir a comunicação em tempo real entre Freelancer e Cliente por meio de um chat integrado à plataforma.

[RF011] – Encerramento do Projeto

Prioridade: Essencial

Descrição: O sistema deve permitir o encerramento do projeto após a conclusão das tarefas e mediante validação do cliente.

2.3. Requisitos Não Funcionais (RNF)**[RNF001] – Usabilidade**

Prioridade: Essencial

Descrição:

A interface do sistema deve ser simples, intuitiva e consistente, permitindo que novos usuários compreendam rapidamente as funcionalidades da plataforma sem necessidade de treinamento extensivo.

[RNF002] – Desempenho

Prioridade: Essencial

Descrição:

A plataforma deve responder às ações do usuário em tempo adequado, garantindo carregamento rápido das telas e fluidez na navegação.

[RNF003] – Segurança

Prioridade: Essencial

Descrição:

O sistema deve garantir autenticação segura dos usuários e proteger dados sensíveis, assegurando que apenas usuários autorizados tenham acesso às funcionalidades e informações da plataforma.

[RNF004] – Confiabilidade

Prioridade: Essencial

Descrição:

O sistema deve operar de forma estável e previsível, evitando falhas frequentes e garantindo consistência das informações armazenadas.

[RNF005] – Manutenibilidade

Prioridade: Importante

Descrição:

O sistema deve ser desenvolvido de forma estruturada e modular, facilitando a manutenção, correção de erros e implementação de novas funcionalidades.

[RNF006] – Portabilidade

Prioridade: Importante

Descrição:

A aplicação deve ser acessível em diferentes dispositivos e ambientes compatíveis, permitindo o uso da plataforma em dispositivos móveis e diferentes sistemas operacionais.

2.4. Modelagem do Banco de Dados

O sistema G Dev Flow utiliza um banco de dados relacional para armazenamento persistente das informações da aplicação.

O Sistema Gerenciador de Banco de Dados (SGBD) adotado no projeto é o *PostgreSQL*, um banco de dados relacional *open-source* amplamente utilizado em aplicações corporativas devido à sua confiabilidade, desempenho, integridade transacional e conformidade com o padrão SQL.

A comunicação entre a aplicação e o banco de dados é realizada através do *Spring Boot* com *Spring Data JPA*, que utiliza o padrão ORM (*Object Relational Mapping*) para mapear as entidades da aplicação em tabelas do banco de dados.

O *PostgreSQL* é responsável por armazenar informações relacionadas às principais entidades do sistema:

- Usuários da plataforma
- Projetos
- Tarefas
- Validações de tarefas

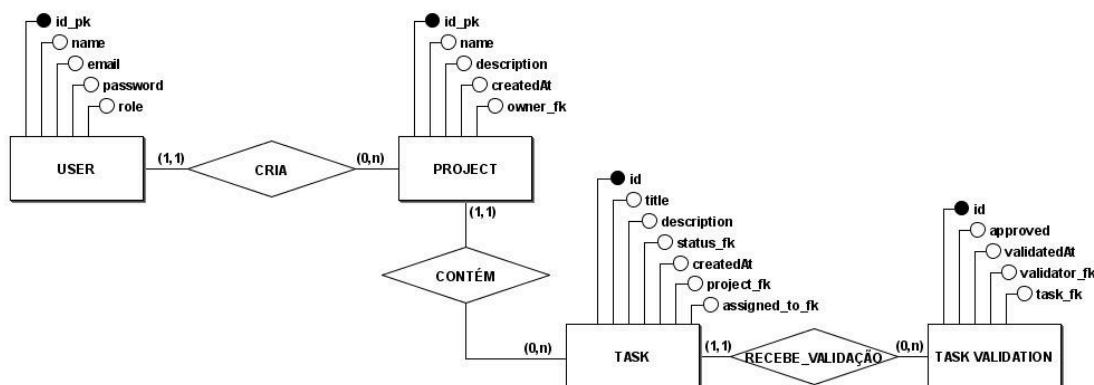
O sistema implementa as operações básicas de CRUD (Create, Read, Update, Delete), permitindo a criação, consulta, atualização e exclusão de registros no banco de dados.

Além disso, o *PostgreSQL* oferece recursos importantes como:

- Controle de integridade referencial por meio de chaves estrangeiras
- Suporte a transações
- Controle de concorrência
- Alta confiabilidade na persistência dos dados

Esses recursos garantem maior consistência e segurança no armazenamento das informações do sistema.

Figura 1 – Modelo Entidade Relacionamento (MER)



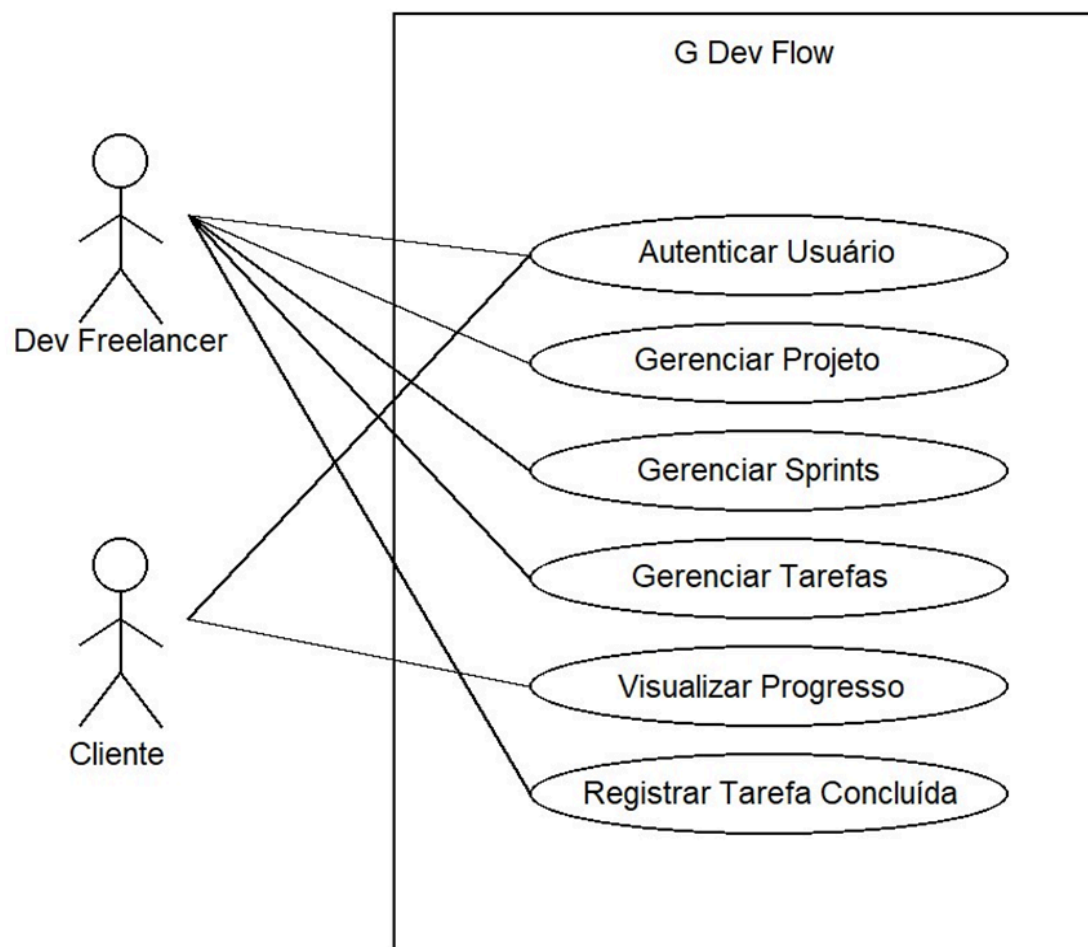
Fonte: Autoria própria, 2026.

2.5. Diagrama de Casos de Uso

Os diagramas de casos de uso são utilizados na fase de análise de requisitos de software para representar, de forma gráfica, as interações entre os atores (usuários) e o sistema de maneira clara e objetiva. De acordo com a documentação da Object Management Group (2014), os casos de uso são utilizados para descrever as funcionalidades de um sistema a partir da perspectiva dos usuários, permitindo a identificação dos requisitos funcionais de forma estruturada.

O principal objetivo do diagrama de casos de uso é identificar os atores envolvidos, que podem ser usuários, sistemas externos ou qualquer outro participante que interage com o software.

Figura 2 – Diagrama de Casos de Uso



Fonte: Autoria própria, 2026

2.5.1. Especificação de Casos de Uso

Tabela 1 - Especificação do Caso de Uso UC1

Campo	Conteúdo
Identificação	UC01 – Autenticar usuário
Atores	Freelancer, Cliente
Descrição breve	Permite que usuários cadastrados acessem a plataforma utilizando credenciais válidas.
Pré-condições	Usuário deve estar previamente cadastrado no sistema.
Pós-condições (sucesso)	Usuário autenticado e acesso liberado conforme seu papel.
Pós-condições (falha)	Acesso negado e usuário permanece na tela de login.
Fluxo principal	1. Usuário acessa tela de login 2. Sistema solicita e-mail e senha 3. Usuário informa credenciais 4. Sistema valida dados 5. Sistema autentica usuário
Fluxo alternativo(s)	Credenciais inválidas → sistema solicita nova tentativa
Regras de negócio	Apenas usuários cadastrados podem acessar o sistema
Exceções	Falha na conexão com banco de dados

Fonte: Autoria própria, 2026

Tabela 2 - Especificação do Caso de Uso UC2

Campo	Conteúdo
Identificação	UC02 – Gerenciar projetos
Atores	Freelancer
Descrição breve	Permite ao freelancer criar, visualizar e encerrar projetos.
Pré-condições	Usuário autenticado como freelancer
Pós-condições (sucesso)	Projeto criado, atualizado ou encerrado com sucesso
Pós-condições (falha)	Operação não realizada
Fluxo principal	1. Freelancer acessa área de projetos 2. Seleciona criar/editar projeto 3. Informa dados do projeto 4. Sistema valida dados 5. Sistema salva projeto
Fluxo alternativo(s)	Dados inválidos → sistema solicita correção
Regras de negócio	Apenas freelancers podem criar e gerenciar projetos
Exceções	Erro ao salvar no banco

Fonte: Autoria própria, 2026

Tabela 3 - Especificação do Caso de Uso UC3

Campo	Conteúdo
Identificação	UC03 – Gerenciar sprints
Atores	Freelancer
Descrição breve	Permite ao freelancer criar e organizar sprints dentro de um projeto.
Pré-condições	Projeto deve existir
Pós-condições (sucesso)	Sprint criada ou atualizada com sucesso
Pós-condições (falha)	Sprint não criada ou atualizada
Fluxo principal	1. Freelancer acessa projeto 2. Seleciona criar sprint 3. Informa nome, datas e status 4. Sistema valida 5. Sistema salva sprint
Fluxo alternativo(s)	Datas inválidas → sistema solicita ajuste
Regras de negócio	Sprint deve estar vinculada a um projeto
Exceções	Falha no banco de dados

Fonte: Autoria própria, 2026

Tabela 4 - Especificação do Caso de Uso UC4

Campo	Conteúdo
Identificação	UC04 – Gerenciar tarefas
Atores	Freelancer
Descrição breve	Permite ao freelancer cadastrar e atualizar tarefas dentro de uma sprint.
Pré-condições	Sprint deve existir
Pós-condições (sucesso)	Tarefa criada ou atualizada com sucesso
Pós-condições (falha)	Tarefa não criada
Fluxo principal	1. Freelancer acessa sprint 2. Cria nova tarefa 3. Informa título, descrição e responsável 4. Sistema valida 5. Sistema salva tarefa
Fluxo alternativo(s)	Dados inválidos → sistema solicita correção
Regras de negócio	Tarefa deve estar vinculada a uma sprint
Exceções	Erro ao persistir dados

Fonte: Autoria própria, 2026

Tabela 5 - Especificação do Caso de Uso UC5

Campo	Conteúdo
Identificação	UC05 – Visualizar progresso do projeto
Atores	Cliente
Descrição breve	Permite ao cliente acompanhar a evolução do projeto.
Pré-condições	Cliente deve estar vinculado ao projeto
Pós-condições (sucesso)	Progresso exibido corretamente
Pós-condições (falha)	Dados não exibidos
Fluxo principal	1. Cliente acessa projeto 2. Sistema exibe progresso (sprints e tarefas) 3. Cliente acompanha evolução
Fluxo alternativo(s)	Projeto inexistente → sistema informa erro
Regras de negócio	Cliente só pode visualizar projetos vinculados a ele
Exceções	Falha ao carregar dados

Fonte: Autoria própria, 2026

Tabela 6 - Especificação do Caso de Uso UC6

Campo	Conteúdo
Identificação	UC06 – Registrar tarefa concluída
Atores	Freelancer
Descrição breve	Permite ao freelancer marcar tarefas como concluídas.
Pré-condições	Tarefa deve existir e estar atribuída ao usuário
Pós-condições (sucesso)	Tarefa marcada como concluída
Pós-condições (falha)	Status não alterado
Fluxo principal	1. Freelancer acessa tarefa 2. Seleciona concluir tarefa 3. Sistema atualiza status 4. Sistema salva alteração
Fluxo alternativo(s)	Tarefa já concluída → sistema informa
Regras de negócio	Apenas o responsável pode concluir a tarefa
Exceções	Falha ao atualizar banco

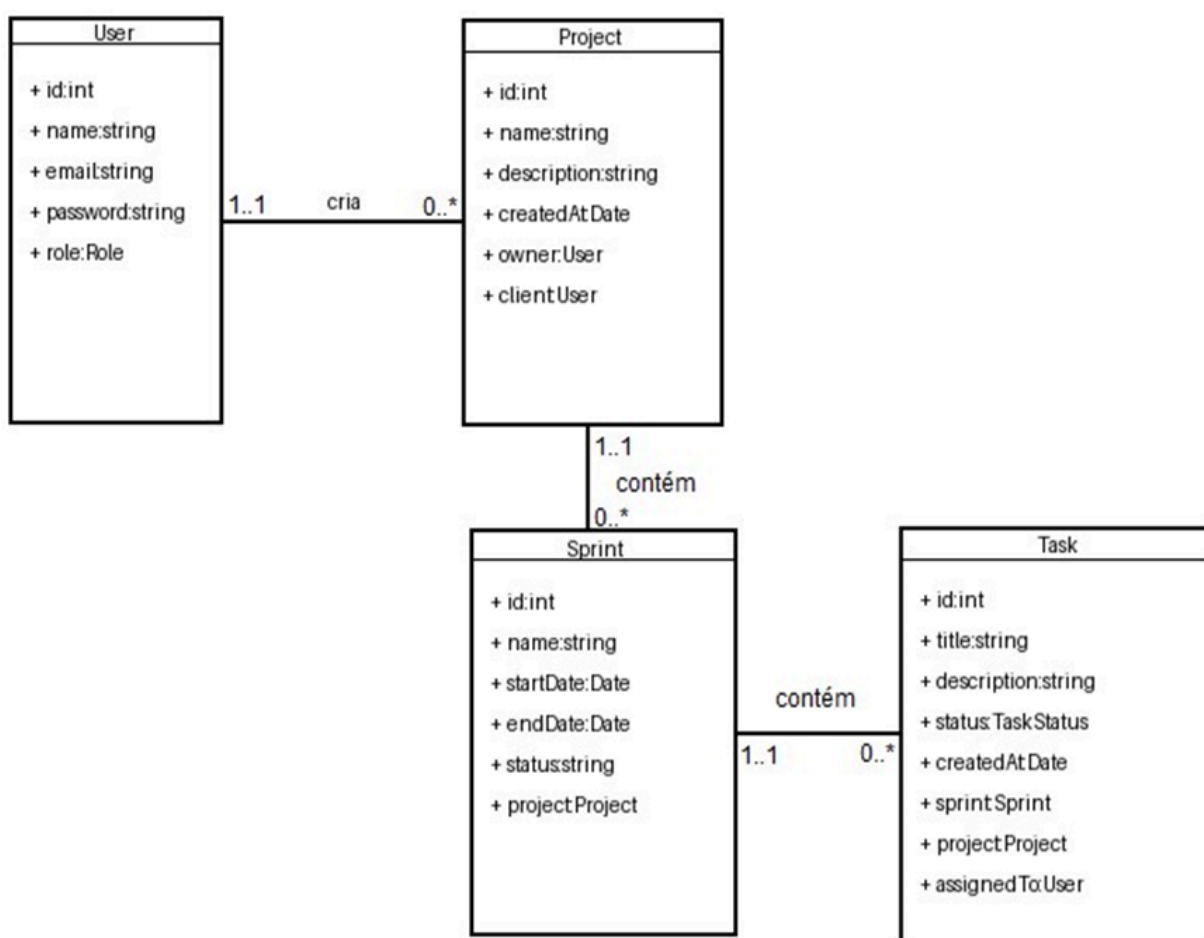
Fonte: Autoria própria, 2026

2.6. Diagrama de Classes

O diagrama de classes é um dos principais diagramas da UML (Unified Modeling Language) e tem como objetivo representar a estrutura estática de um sistema orientado a objetos. Ele descreve as classes do sistema, seus atributos, métodos e os relacionamentos existentes entre elas, permitindo visualizar como os elementos do sistema estão organizados e como se relacionam.

De acordo com a documentação da Object Management Group (2014), o diagrama de classes descreve a estrutura estática de um sistema, incluindo classes, atributos, operações e os relacionamentos entre esses elementos, sendo uma das principais formas de representar a organização interna de um software.

Figura 3 - Diagrama de Classes



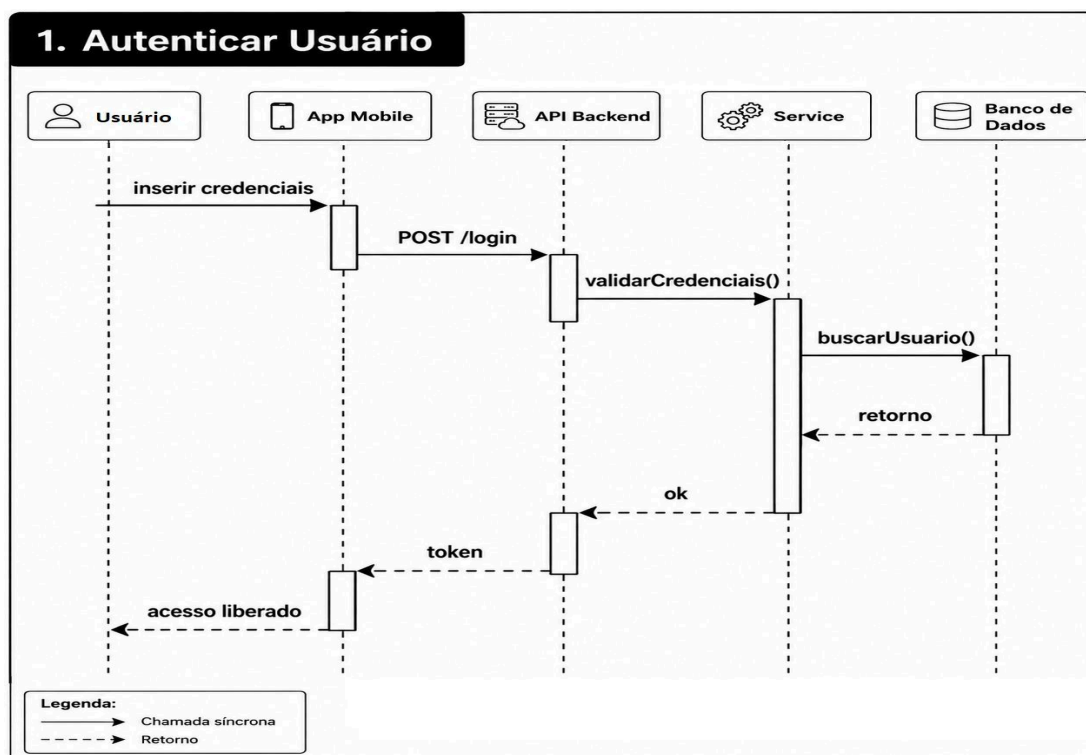
Fonte: Autoria própria, 2026.

2.7. Diagrama de Sequência

O diagrama de sequência é um tipo de diagrama da UML (Unified Modeling Language) utilizado para representar a interação entre objetos ou componentes de um sistema ao longo do tempo. Ele mostra a ordem em que as mensagens são trocadas entre os elementos do sistema para realizar determinada funcionalidade ou processo.

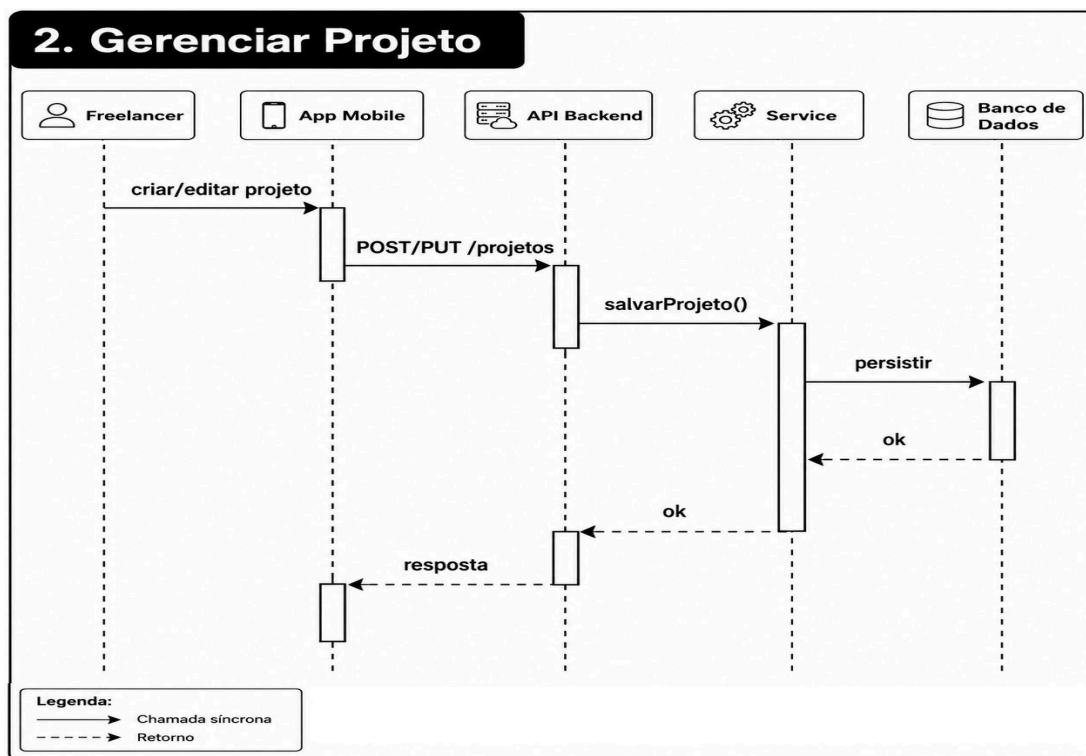
De acordo com a documentação da Object Management Group (2014), os diagramas de sequência são utilizados para descrever interações entre elementos de um sistema por meio da troca de mensagens organizadas em uma sequência temporal, permitindo compreender o comportamento dinâmico do software.

Figura 4 – Diagrama de Sequência - Autenticar Usuário



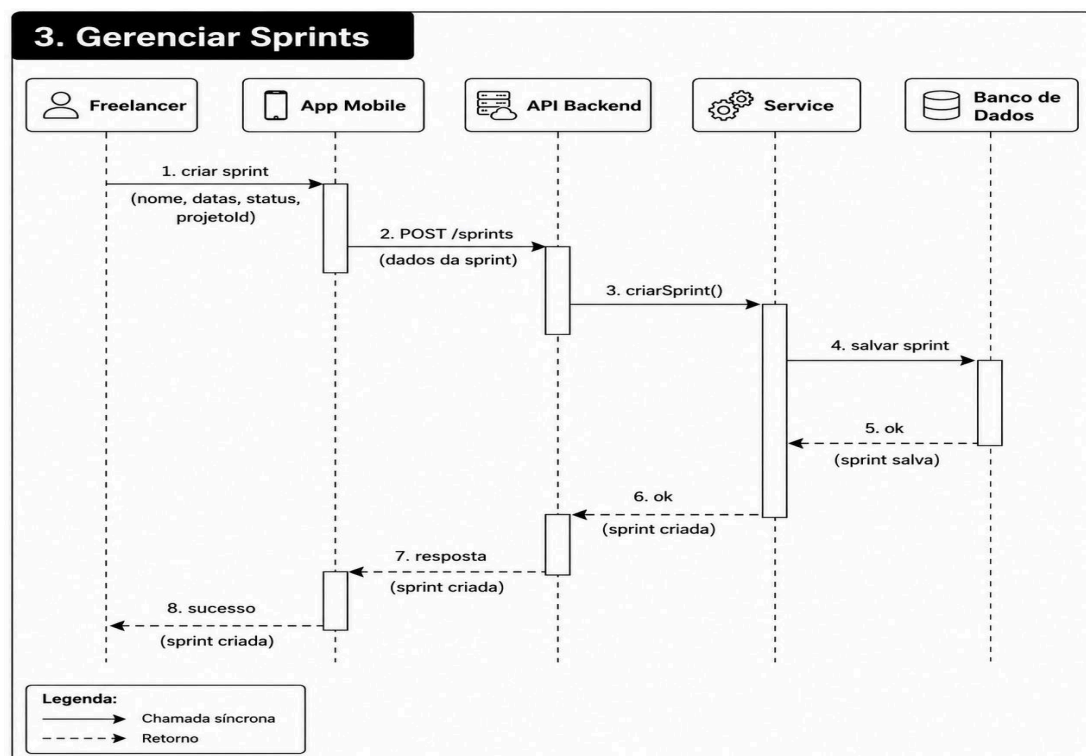
Fonte: Autoria própria, 2026.

Figura 5 – Diagrama de Sequência - Gerenciar Projeto



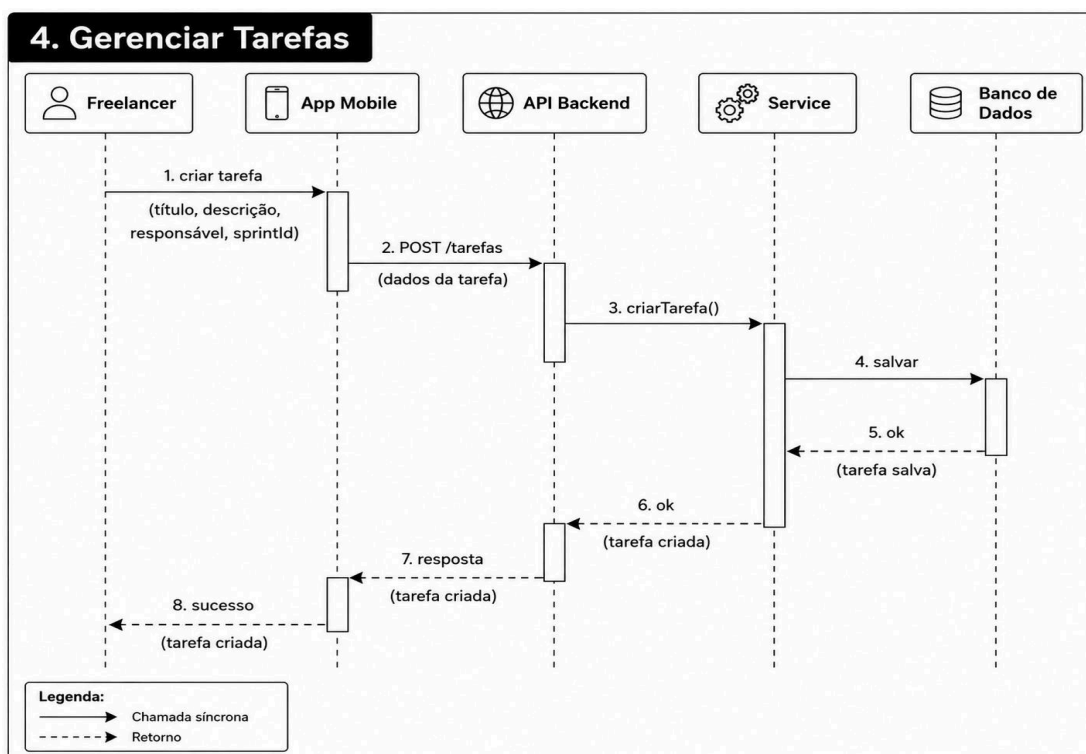
Fonte: Autoria própria, 2026.

Figura 6 – Diagrama de Sequência - Gerenciar Sprints



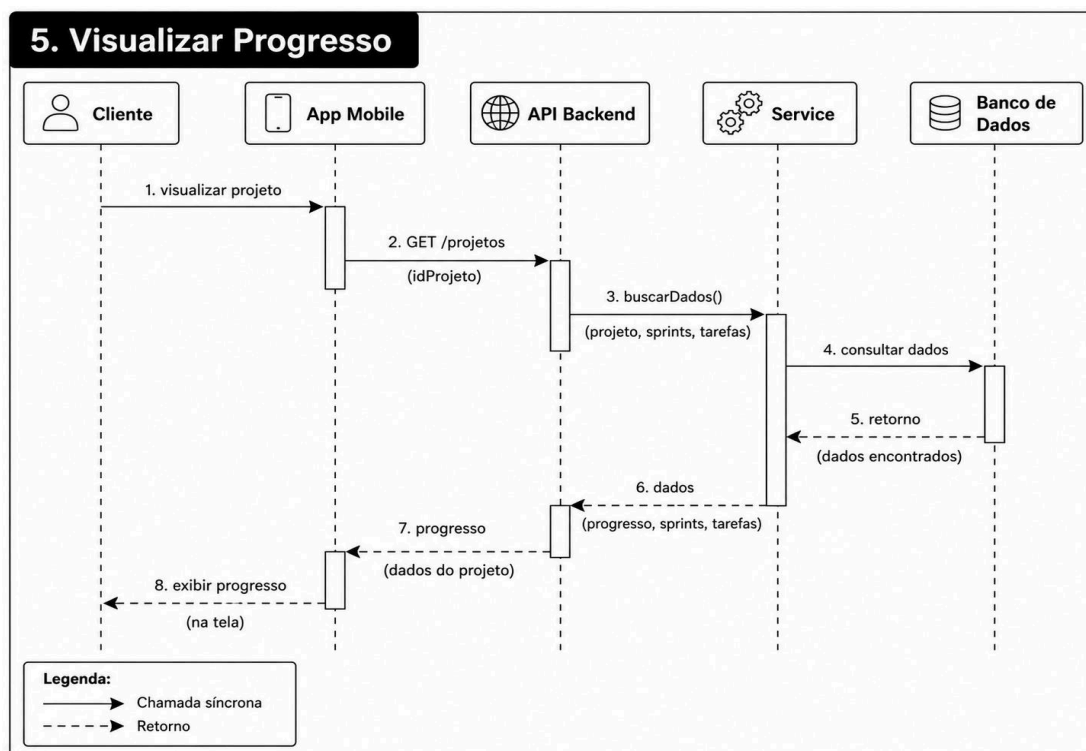
Fonte: Autoria própria, 2026.

Figura 7 – Diagrama de Sequência - Gerenciar Tarefas



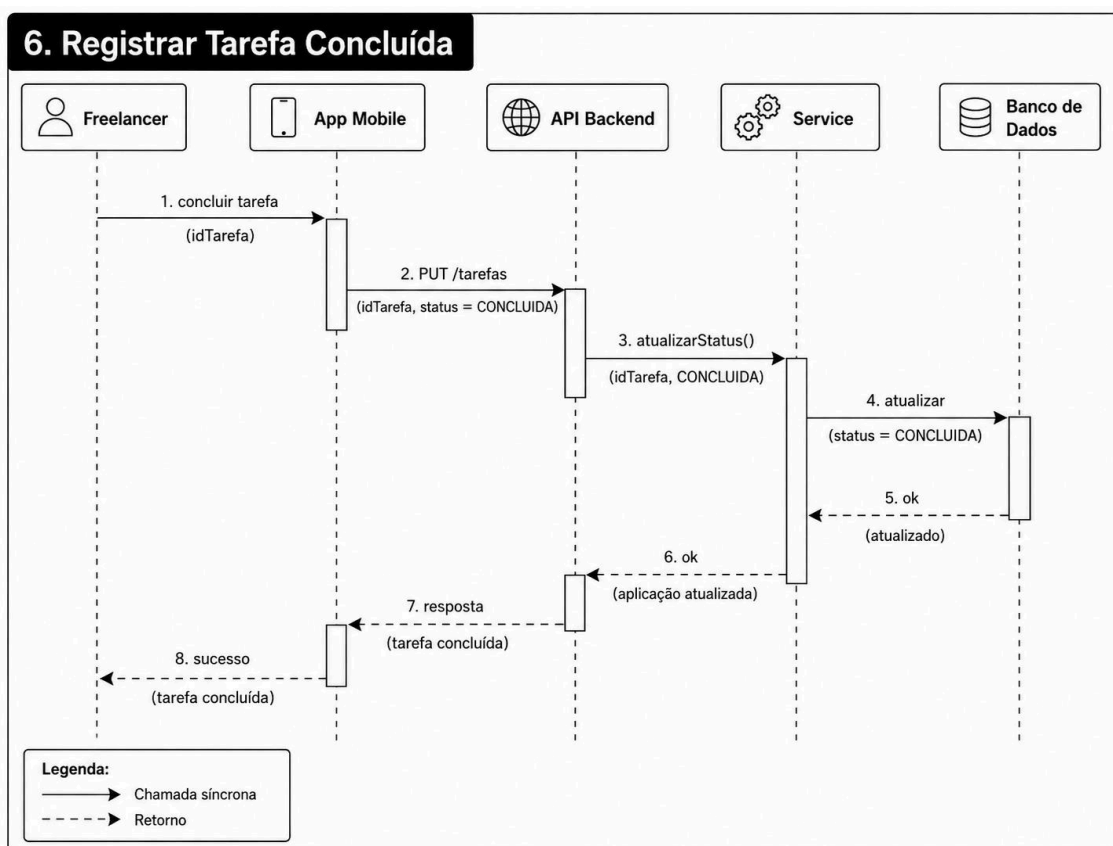
Fonte: Autoria própria, 2026.

Figura 8 – Diagrama de Sequência - Visualizar Progresso



Fonte: Autoria própria, 2026.

Figura 9 – Diagrama de Sequência - Registrar Tarefa Concluída



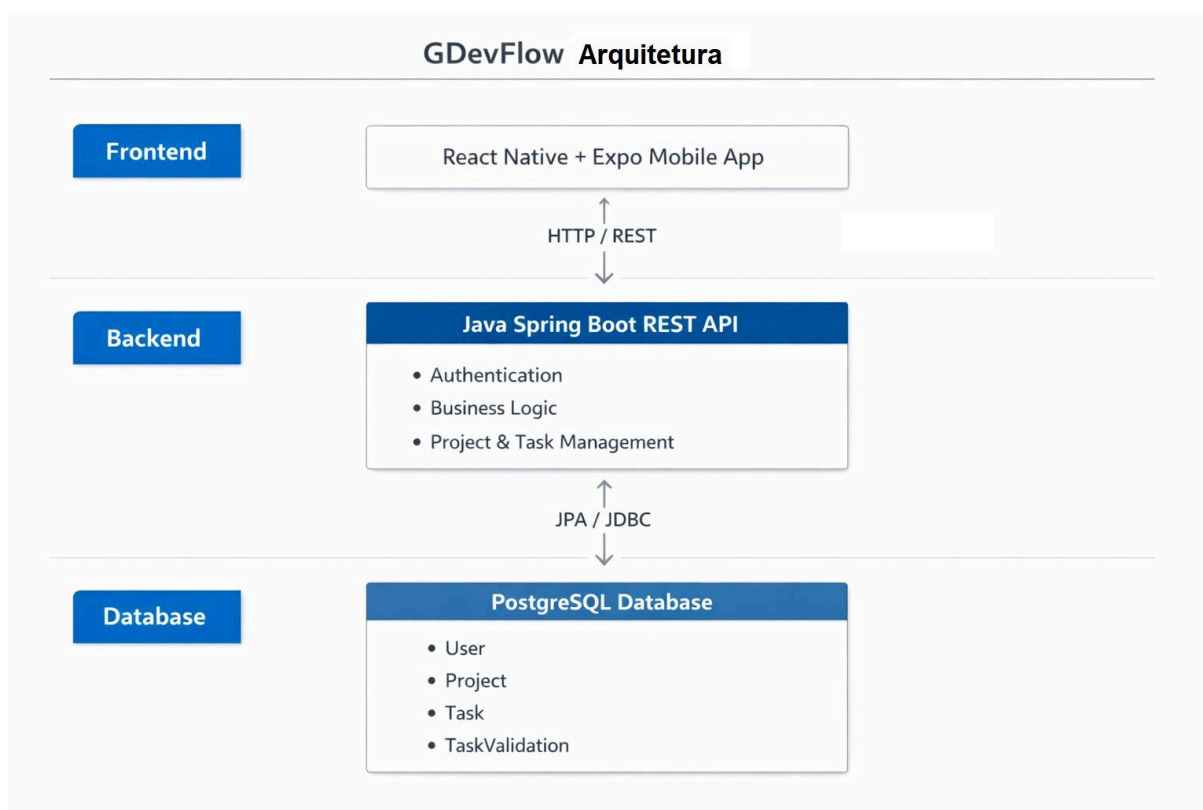
Fonte: Autoria própria, 2026.

2.8. Diagrama de Arquitetura

O diagrama de arquitetura de software é utilizado para representar, de forma visual, a estrutura geral de um sistema, demonstrando seus principais componentes, módulos e a forma como eles se comunicam. Esse tipo de diagrama auxilia na compreensão da organização do sistema, evidenciando as camadas, tecnologias utilizadas e as interações entre os diferentes elementos da aplicação.

Segundo Ian Sommerville (2011), a arquitetura de software descreve a estrutura fundamental de um sistema, identificando seus componentes e as relações entre eles.

Figura 10 – Diagrama de Arquitetura



Fonte: Autoria própria, 2026

3 IMPLEMENTAÇÃO

Esta seção apresenta como o sistema foi desenvolvido, descrevendo suas principais características técnicas e os recursos utilizados durante a construção da solução. Também são incluídas informações sobre os testes realizados e as telas que compõem a aplicação.

3.1 Descrição da Aplicação

O sistema G Dev Flow foi implementado utilizando uma arquitetura baseada no modelo cliente-servidor, estruturada em três camadas principais: frontend, backend e banco de dados, conforme ilustrado na Figura 11. Essa abordagem visa promover a separação de responsabilidades, facilitando a manutenção, escalabilidade e evolução da aplicação.

A camada de frontend foi desenvolvida utilizando React Native com Expo, permitindo a criação de uma aplicação mobile multiplataforma. O React Native possibilita o desenvolvimento de interfaces utilizando JavaScript e componentes reutilizáveis, enquanto o Expo fornece ferramentas que simplificam o processo de desenvolvimento, testes e execução da aplicação, aumentando a produtividade e reduzindo a complexidade do desenvolvimento mobile (EXPO, 2024).

A comunicação entre o frontend e o backend ocorre por meio de requisições HTTP, seguindo o padrão REST (Representational State Transfer), que define boas práticas para a construção de APIs escaláveis e desacopladas. Conforme definido por Fielding (2000), o estilo arquitetural REST promove simplicidade, escalabilidade e independência entre cliente e servidor.

O backend foi implementado em Java utilizando o framework Spring Boot, responsável pelo processamento das regras de negócio do sistema. O Spring Boot oferece suporte à criação de APIs REST, gerenciamento de dependências e configuração automática da aplicação, facilitando o desenvolvimento de sistemas web modernos (SPRING, 2023).

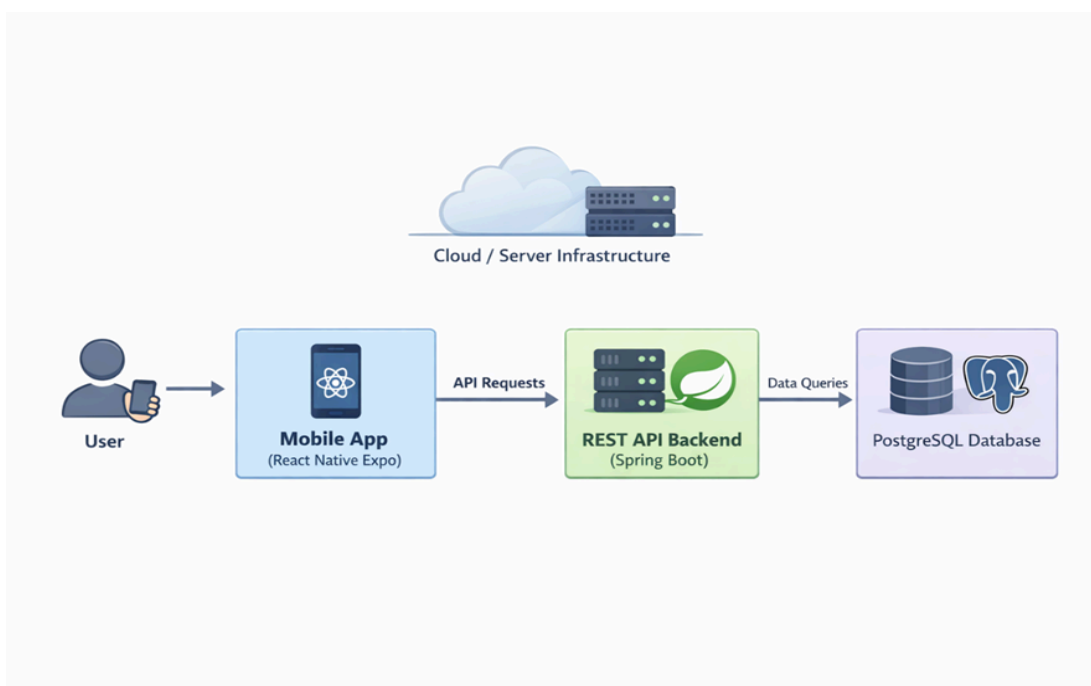
No backend, foi adotado o padrão arquitetural baseado em Controller, Service e Repository, no qual:

- Controller é responsável por receber e responder às requisições HTTP;
- Service concentra as regras de negócio da aplicação;
- Repository realiza a comunicação com o banco de dados.

Essa separação reduz o acoplamento entre componentes e melhora a legibilidade e manutenibilidade do código.

Para a persistência dos dados, foi utilizado o sistema gerenciador de banco de dados PostgreSQL, escolhido por sua confiabilidade, suporte a transações e aderência ao modelo relacional. A integração com o backend é realizada por meio de JPA (Java Persistence API) e Hibernate, utilizando o padrão ORM (Object-Relational Mapping), que permite mapear objetos da aplicação para tabelas no banco de dados, reduzindo a necessidade de consultas SQL complexas e aumentando a produtividade (HIBERNATE, 2023).

Figura 11 – Ecossistema



Fonte: Autoria própria, 2026.

3.2 Implantação

A aplicação G Dev Flow foi executada e validada em ambiente local de desenvolvimento, com o objetivo de realizar os testes funcionais e verificar o funcionamento das principais funcionalidades do sistema. Para isso, foram utilizados o backend em Java com Spring Boot, o banco de dados PostgreSQL e a aplicação mobile desenvolvida em React Native com Expo. Nesta etapa do projeto, o sistema não foi publicado em loja de aplicativos nem implantado em ambiente de nuvem, ficando sua disponibilização em produção como uma possibilidade de evolução futura.

3.3 Roteiro de Testes

Os roteiros de testes foram elaborados com o objetivo de validar o funcionamento das principais funcionalidades do sistema G Dev Flow. Esses testes foram definidos com base nos requisitos levantados durante o desenvolvimento, buscando garantir que cada funcionalidade atenda ao comportamento esperado em diferentes cenários de uso.

A organização dos testes foi realizada por meio de uma tabela, na qual estão descritos os casos de teste, incluindo etapas de execução, dados de entrada, ações realizadas e resultados esperados. Essa estrutura permite um acompanhamento sistemático do processo de validação, facilitando a identificação de falhas e inconsistências.

Os testes contemplam funcionalidades essenciais do sistema, como autenticação de usuários, gerenciamento de projetos, criação de sprints e controle de tarefas, assegurando o correto funcionamento de cada componente. A aplicação desses roteiros contribuiu para a melhoria da experiência do usuário e para a garantia da qualidade final da aplicação.

Para fins de consulta e rastreabilidade, o repositório do projeto pode ser acessado em [nosso repositório](#).

Tabela 7 - Roteiro de Testes UC1 - Autenticar Usuário

Procedimento: Necessário ter acesso ao sistema como usuário					
Fluxo básico: UC1 - Autenticar Usuário					
ID	Passos para execução	Dados de entrada	Resultado esperado	Resultado verificado	Observações
1	Abrir o aplicativo na tela de login	e-mail: thibor@magnani.com senha: 12345678	Digitar um e-mail e senha válidos de usuário para verificar se ao clicar em "Entrar" será direcionado para a tela de início.	OK	
2	Abrir o aplicativo na tela de login	e-mail: thibor@magnani.com senha: 12378	Verificar se ao digitar o e-mail ou a senha incorretamente, é exibido o aviso "E-mail ou senha inválidos".	OK	
3	Abrir o aplicativo na tela de login		Não preencher nenhum campo e clicar no botão "Entrar" para verificar se serão exibidas as mensagens de que os campos são de preenchimento obrigatório.	OK	
4	Abrir o aplicativo na tela de login		Preencher o campo "E-mail" mas deixar o campo "Senha" em branco e clicar no botão "Entrar" para verificar se é exibida a mensagem para informar a senha	OK	
5	Abrir o aplicativo na tela de login		Preencher o campo "Senha" mas deixar o campo "E-mail" em branco e clicar no botão "Entrar" para verificar se é exibida a mensagem para informar o Email.	OK	
6	Abrir o aplicativo na tela de login e clicar em "Não tem uma conta? Cadastre-se".		Verificar se no rodapé a opção "Criar conta" direciona para a tela de cadastro.	OK	
7	Estar na tela de cadastro		Clicar em "Criar conta" sem preencher nenhuma informação, a fim de verificar se as mensagens de informar os campos estão sendo mostradas	OK	
8	Estar na tela de cadastro	email: thibor	Preencher todos os campos corretamente, com exceção do email e clicar em "Criar conta", para verificar se a mensagem de digitar um email válido aparece.	OK	
9	Estar na tela de cadastro	senha: 1234567	Preencher todos os campos corretamente, porém digitar uma senha com menos de 8 caracteres e clicar em "Criar conta" a fim de verificar se o cadastro não é concluído e a mensagem de tamanho de senha mínimo é mostrada	OK	
10	Estar na tela de cadastro	senha: 12345678 confirmação de senha: 12345678A	Preencher todos os campos corretamente, porém digitar uma confirmação de senha diferente da senha digitada no primeiro campo, a fim de verificar se o aviso de que as senhas não coincidem é mostrado	OK	

11	Estar na tela de cadastro	nome: Thibor email: thibor@magnani.com senha: 12345678 confirmar senha: 12345678	Preencher todos os campos corretamente e clicar em "Criar conta", para verificar se o usuário é criado e se ele já é redirecionado automaticamente para a senha de login	OK	
----	---------------------------	--	--	----	--

Fonte: Autoria própria, 2026

Tabela 8 - Roteiro de Testes UC2 - Gerenciar Projetos

Procedimento: Necessário ter acesso ao sistema autenticado como FREELANCER para execução do CRUD de Projetos e CRUD de Sprints.					
Fluxo básico: UC02 – Gerenciar Projetos					
ID	Passos para execução	Dados de entrada	Resultado esperado	Resultado verificado	Observações
12	Acessar a tela de projetos após login como freelancer	Credenciais válidas de freelancer	A tela deve listar os projetos do usuário autenticado, exibindo nome, descrição e status quando existirem.	OK	
13	Criar um novo projeto com preenchimento válido	Nome: G DEV FLOW Mobile Descrição: App para gestão de projetos	O sistema deve cadastrar o projeto com sucesso e exibi-lo na listagem.	OK	
14	Criar projeto sem preencher o nome	Nome em branco Descrição preenchida	O sistema deve bloquear o cadastro e informar que o nome é obrigatório.	OK	
15	Criar projeto sem preencher a descrição	Nome preenchido Descrição em branco	O sistema deve bloquear o cadastro e informar que a descrição é obrigatória na interface.	OK	
16	Criar dois projetos diferentes em sequência	Projeto A e Projeto B com dados válidos	Ambos devem ser cadastrados corretamente e aparecer na listagem sem sobrescrever dados.	OK	
17	Abrir os detalhes de um projeto existente	Selecionar um projeto já cadastrado	O sistema deve exibir os dados do projeto, métricas e área de sprints vinculadas.	OK	
18	Atualizar nome e descrição de um projeto	Novo nome e nova descrição válidos	O sistema deve salvar as alterações e refletir os novos dados na listagem e na tela de detalhes.	OK	
19	Tentar atualizar projeto com nome vazio	Editar projeto removendo o nome	O sistema deve impedir a atualização e exibir validação do campo obrigatório.	OK	
20	Realizar atualização de projeto e voltar para a listagem	Projeto previamente cadastrado	A listagem deve carregar os dados atualizados sem inconsistência visual ou duplicidade.	OK	
21	Acessar a área de sprints de um projeto sem sprints cadastradas	Projeto recém-criado	A tela deve exibir estado vazio orientando a criação da primeira sprint.	OK	

Fonte: Autoria própria, 2026

Tabela 9 - Roteiro de Testes UC3 - Gerenciar Sprints

Procedimento: Necessário ter acesso ao sistema autenticado como FREELANCER para execução do CRUD de Projetos e CRUD de Sprints.					
Fluxo básico: UC03 – Gerenciar Sprints					
ID	Passos para execução	Dados de entrada	Resultado esperado	Resultado verificado	Observações
22	Criar nova sprint vinculada a um projeto existente	Nome: Sprint 1 Início: 2026-04-15 Fim: 2026-04-22 Status: PLANNED	O sistema deve cadastrar a sprint e exibi-la na seção de sprints do projeto.	OK	
23	Criar sprint sem nome	Nome em branco Datas válidas	O sistema deve bloquear o cadastro e informar que o nome da sprint é obrigatório.	OK	
24	Criar sprint com data final anterior à data inicial	Início: 2026-04-22 Fim: 2026-04-15	O sistema deve impedir o salvamento e informar período inválido.	OK	
25	Criar duas sprints para o mesmo projeto	Sprint 1 e Sprint 2 com períodos válidos	As duas sprints devem permanecer vinculadas ao mesmo projeto e aparecer na lista corretamente.	OK	
26	Abrir uma sprint existente para edição	Selecionar sprint já cadastrada	O sistema deve carregar nome, datas e status atuais da sprint.	OK	
27	Editar nome, período e status da sprint	Atualizar para IN_PROGRESS e alterar datas	O sistema deve persistir as alterações e refletir os novos dados na tela do projeto.	OK	
28	Concluir sprint alterando status para DONE	Sprint existente com dados válidos	O sistema deve salvar o novo status e atualizar os indicadores apresentados no projeto.	OK	
29	Listar sprints de um projeto autenticado	Projeto com sprints cadastradas	A API e a interface devem retornar somente as sprints daquele projeto, ordenadas corretamente.	OK	
30	Validar atualização dos indicadores da tela do projeto após criar e concluir sprints	Projeto com sprints em diferentes status	A tela de detalhes deve recalcular quantitativos e progresso sem apresentar erro visual.	OK	

Fonte: Autoria própria, 2026

Tabela 10 - Roteiro de Testes UC4 - Gerenciar Tarefas

Procedimento: Necessário ter acesso ao sistema autenticado como FREELANCER para execução do CRUD de tarefas e visualização do progresso do projeto.					
Fluxo básico: UC04 – Gerenciar tarefas					
ID	Passos para execução	Dados de entrada	Resultado esperado	Resultado verificado	Observações
31	Criar tarefa válida	Título, descrição, sprint válida	Tarefa criada com sucesso	OK	
32	Criar tarefa sem título	Título vazio	Sistema bloqueia e valida campo obrigatório	OK	
33	Criar tarefa com data limite inválida	DueDate anterior a data de criação	Sistema bloqueia e informa erro	OK	
34	Criar tarefa com data limite posterior à data limite da Sprint	DueDate fora do período da sprint	Sistema bloqueia e informa erro	OK	
35	Criar tarefa com dependência válida	Dependência de tarefa existente	Sistema vincula dependência corretamente	OK	
36	Concluir tarefa sem dependências	Status DONE	Tarefa é concluída com sucesso	OK	
37	Concluir tarefa com dependência pendente	Dependência NÃO concluída	Sistema bloqueia conclusão	OK	
38	Concluir tarefa com dependência concluída	Dependência DONE	Sistema permite conclusão	OK	

Fonte: Autoria própria, 2026

Tabela 11 - Roteiro de Testes UC5 - Visualizar Progresso do Projeto

Procedimento: Necessário ter acesso ao sistema autenticado como FREELANCER para execução do CRUD de tarefas e visualização do progresso do projeto.					
Fluxo básico: UC05 – Visualizar progresso do projeto					
ID	Passos para execução	Dados de entrada	Resultado esperado	Resultado verificado	Observações
39	Listar tarefas da sprint	Sprint existente	Sistema retorna lista correta	OK	
40	Atualizar status da tarefa	Alterar para IN_PROGRESS	Sistema atualiza status corretamente	OK	
41	Visualizar progresso do projeto	Projeto com tarefas	Sistema exibe porcentagem correta	OK	
42	Validar progresso com tarefas concluídas	Parte DONE	Progresso proporcional exibido	OK	
43	Validar progresso sem tarefas	Projeto vazio	Progresso 0%	OK	
44	Realizar cadastro com usuário Cliente	email:gopaulon@mag nanilobo.com senha: 12345678	Sistema cadastra usuário com perfil CLIENT	OK	
45	Acessar painel do cliente sem projetos vinculados	Usuário CLIENT sem projetos	Sistema exibe mensagem informando que não há projetos vinculados	OK	
46	Acessar painel do cliente com projetos vinculados	Usuário CLIENT com projetos	Sistema exibe projetos vinculados ao usuário	OK	
47	Cliente abre tela de progresso de projeto	Projeto vinculado ao usuário CLIENT	Sistema exibe progresso do projeto escolhido	OK	
48	Validar que o cliente não vê ações de gerenciamento	Usuário CLIENT na tela de progresso	Botões de criar, editar ou excluir sprints e tarefas não são exibidos	OK	

Fonte: Autoria própria, 2026

Tabela 12 - Roteiro de Testes UC6 - Registrar Tarefa Concluída

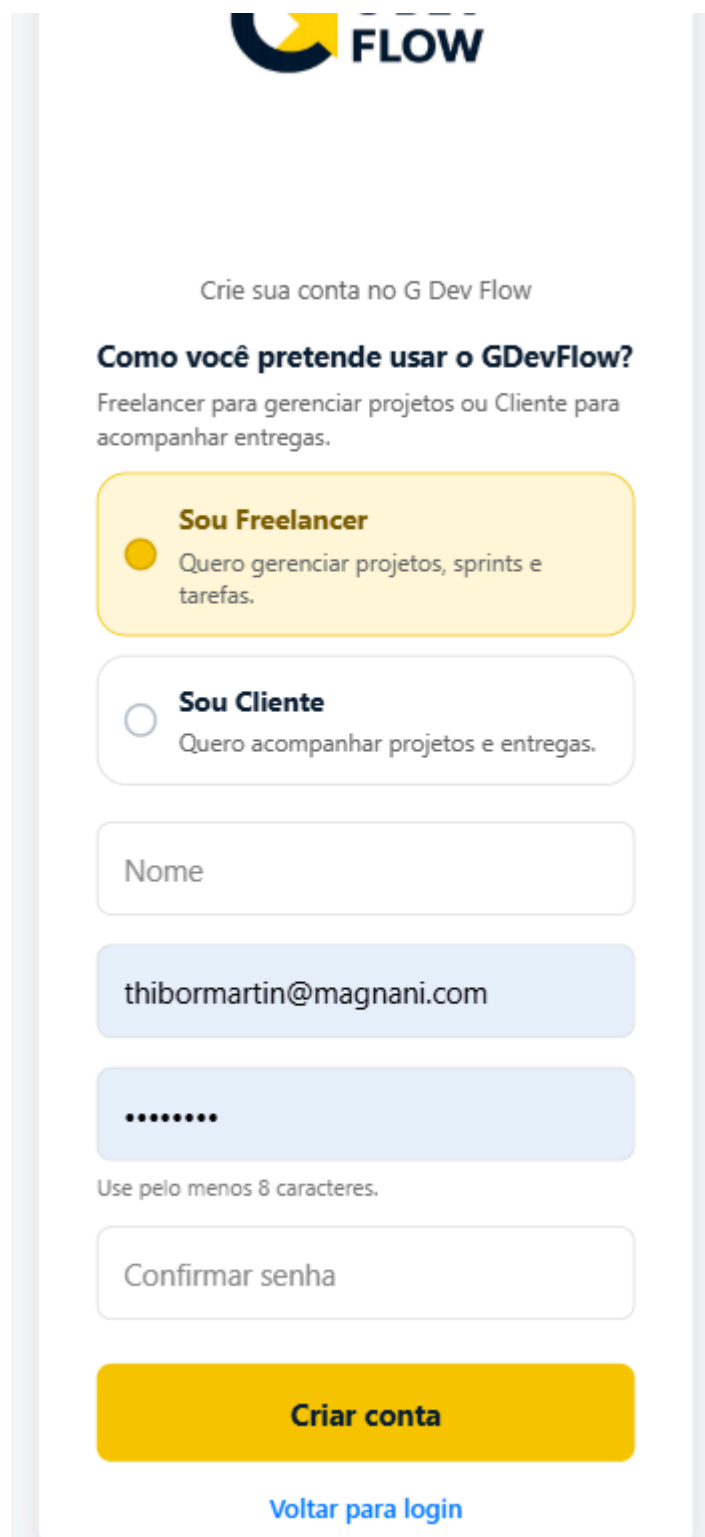
Procedimento: Necessário ter acesso ao sistema autenticado como FREELANCER para execução do CRUD de tarefas e visualização do progresso do projeto.					
Fluxo básico: UC06 - Registrar tarefa concluída					
ID	Passos para execução	Dados de entrada	Resultado esperado	Resultado verificado	Observações
49	Freelancer marcar tarefa como concluída	Task vinculada a projeto do freelancer	Sistema altera o status da tarefa para concluída	OK	
50	Freelancer solicitar aprovação com tarefas pendentes	Projeto com tarefas não concluídas	Sistema bloqueia a solicitação de aprovação	OK	
51	Freelancer solicitar aprovação sem nada inválido	Projeto com tarefas concluídas e cliente vinculado	Sistema envia solicitação ao cliente e muda status para aguardando aprovação do cliente	OK	
52	Cliente visualiza projeto aguardando aprovação	Projeto vinculado ao cliente em status de aguardando aprovação	Sistema exibe botão para aprovar a conclusão do projeto	OK	
53	Cliente aprova conclusão do projeto	Projeto vinculado, 100% concluído e aguardando aprovação	Sistema altera status do projeto para concluído	OK	
54	Verificar projeto concluído no painel do cliente	Projeto com status concluído	Sistema exibe o projeto como concluído	OK	
55	Verificar projeto concluído no painel do freelancer	Projeto aprovado pelo cliente	Sistema exibe status completo para o freelancer	OK	

Fonte: Autoria própria, 2026

3.4 Telas

Na Figura 12 é apresentada a tela de cadastro de usuário no sistema

Figura 12 - Tela - Cadastro





Crie sua conta no G Dev Flow

Como você pretende usar o GDevFlow?

Freelancer para gerenciar projetos ou Cliente para acompanhar entregas.

☒ **Sou Freelancer**
Quero gerenciar projetos, sprints e tarefas.

☐ **Sou Cliente**
Quero acompanhar projetos e entregas.

Nome

thibormartin@magnani.com

.....

Use pelo menos 8 caracteres.

Confirmar senha

Criar conta

[Voltar para login](#)

Fonte: Autoria própria, 2026.

Na Figura 13 é apresentada a tela de login de usuário no sistema

Figura 13 – Tela - Autenticar usuário

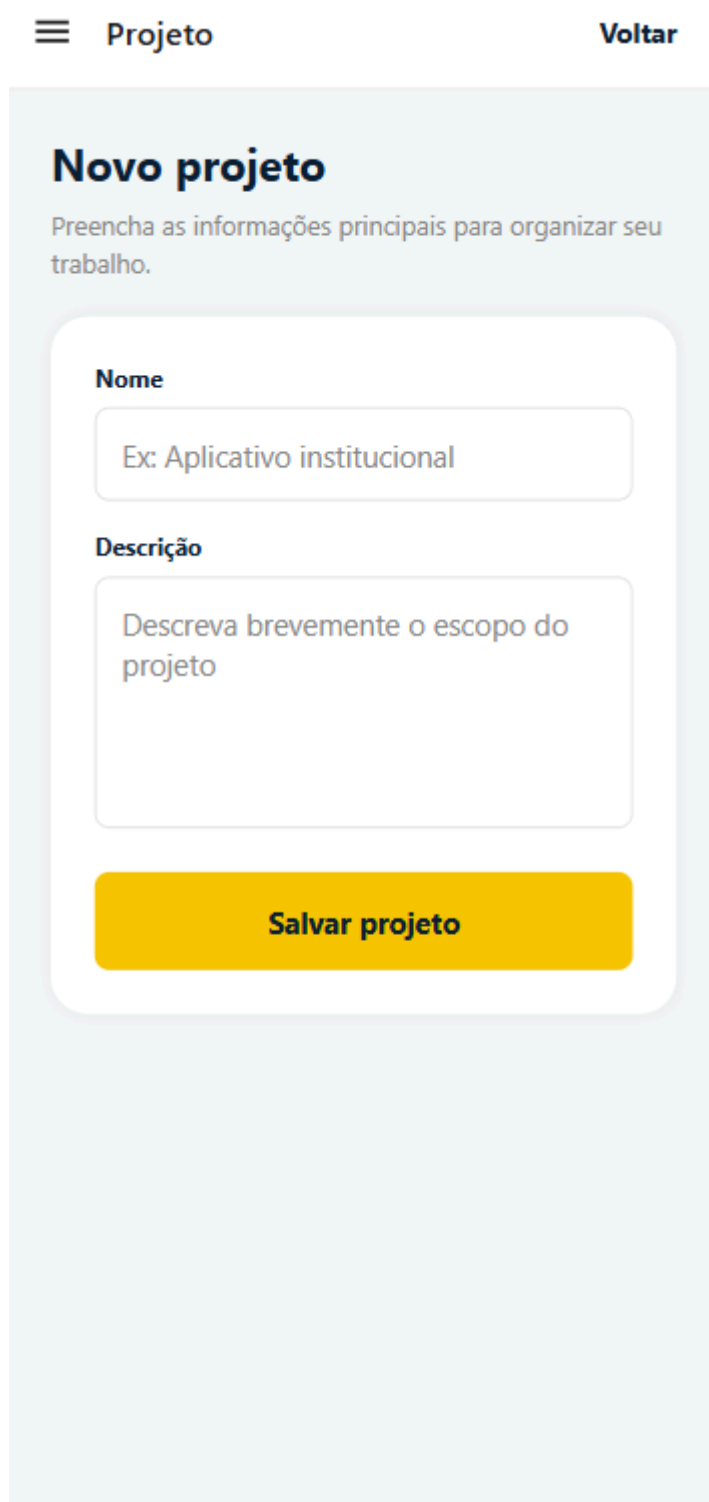


The image shows a mobile application login screen for 'G DEV FLOW'. At the top is the logo, which consists of a stylized 'G' with a yellow arrow pointing right, followed by the text 'G DEV FLOW'. Below the logo is the instruction 'Acesse sua conta para continuar'. There are two input fields: the first contains the email 'thibormartin@magnani.com' and the second contains masked characters '.....'. A large yellow button labeled 'Entrar' is positioned below the password field. Underneath this button is the text 'ou continue com'. There are two more buttons: 'Entrar com Google' and 'Entrar com GitHub'. At the bottom, there is a blue link that says 'Criar conta'.

Fonte: Autoria própria, 2026

Na Figura 14 é apresentada a tela de criação de de um novo projeto

Figura 14 - Tela - Criar projeto



A interface de usuário para criar um novo projeto. No topo, há uma barra de navegação com um ícone de menu hambúrguer à esquerda, o texto "Projeto" no centro e o texto "Voltar" à direita. Abaixo, o título "Novo projeto" é exibido em uma fonte grande e bold. Segue-se uma instrução: "Preencha as informações principais para organizar seu trabalho." O formulário principal contém dois campos de entrada: "Nome" com o exemplo "Ex: Aplicativo institucional" e "Descrição" com o placeholder "Descreva brevemente o escopo do projeto". Um botão amarelo com o texto "Salvar projeto" está localizado na base do formulário.

Projeto Voltar

Novo projeto

Preencha as informações principais para organizar seu trabalho.

Nome

Ex: Aplicativo institucional

Descrição

Descreva brevemente o escopo do projeto

Salvar projeto

Fonte: Autoria própria, 2026.

Na Figura 15 é apresentada a tela de detalhes do projeto

Figura 15 - Tela - Gerenciar projeto



Fonte: Autoria própria, 2026.

Na Figura 16 é apresentada a tela de criação de sprint

Figura 16 - Tela - Criar sprint

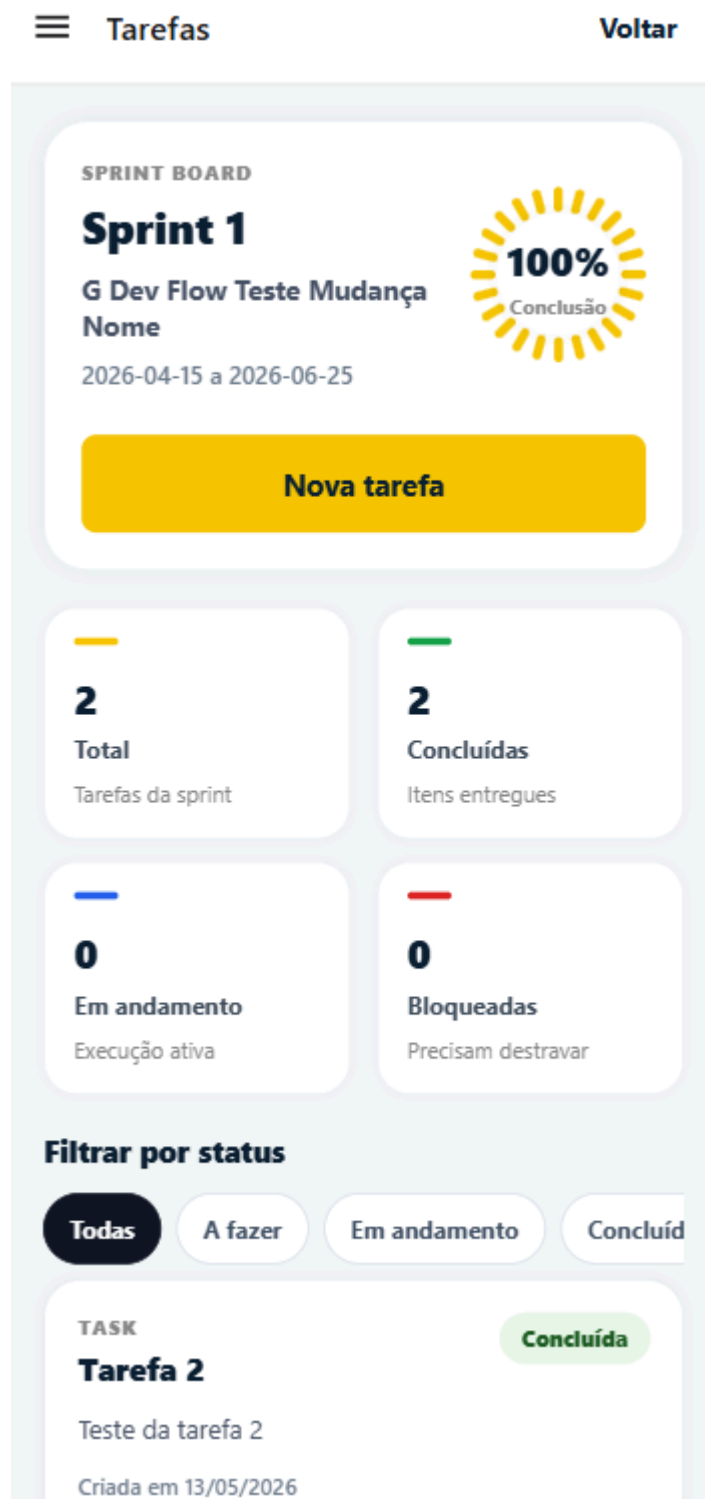
A interface de criação de uma nova sprint é exibida em um layout limpo. No topo, há uma barra de navegação com um ícone de menu hambúrguer à esquerda, o título "Sprint" no centro e um botão "Voltar" à direita. O formulário principal, intitulado "Nova sprint", contém as seguintes seções:

- Nome:** Um campo de texto com o exemplo "Ex: Sprint 1".
- Descrição:** Um campo de texto com o placeholder "Descreva o objetivo da sprint".
- Data de início:** Um campo de data com o valor "30/05/2026" e um botão "Selecionar" em azul.
- Data de fim:** Um campo de data com o valor "30/05/2026" e um botão "Selecionar" em azul.
- Status:** Três opções de status em botões: "Planejada" (destacado em amarelo), "Em andamento" e "Concluída".

Fonte: Autoria própria, 2026.

Na Figura 17 é apresentada a tela de gerenciamento de sprint(sprint board)

Figura 17 - Tela - Gerenciar sprint



Fonte: Autoria própria, 2026.

Na Figura 18 é apresentada a tela de criação de tarefa

Figura 18 – Tela - Criar tarefa

Tarefa **Voltar**

Editar tarefa

Organize o backlog com data limite, dependências e status da entrega.

Projeto vinculado

G Dev Flow Teste Mudança Nome

Sprint selecionada

Sprint 1

Título

Tarefa 2

Descrição

Teste da tarefa 2

Data limite

21/05/2026 **Selecionar**

Responsável (opcional)

Ex: Gabriel

Fonte: Autoria própria, 2026.

Na Figura 19 é apresentada a tela de conclusão de tarefa

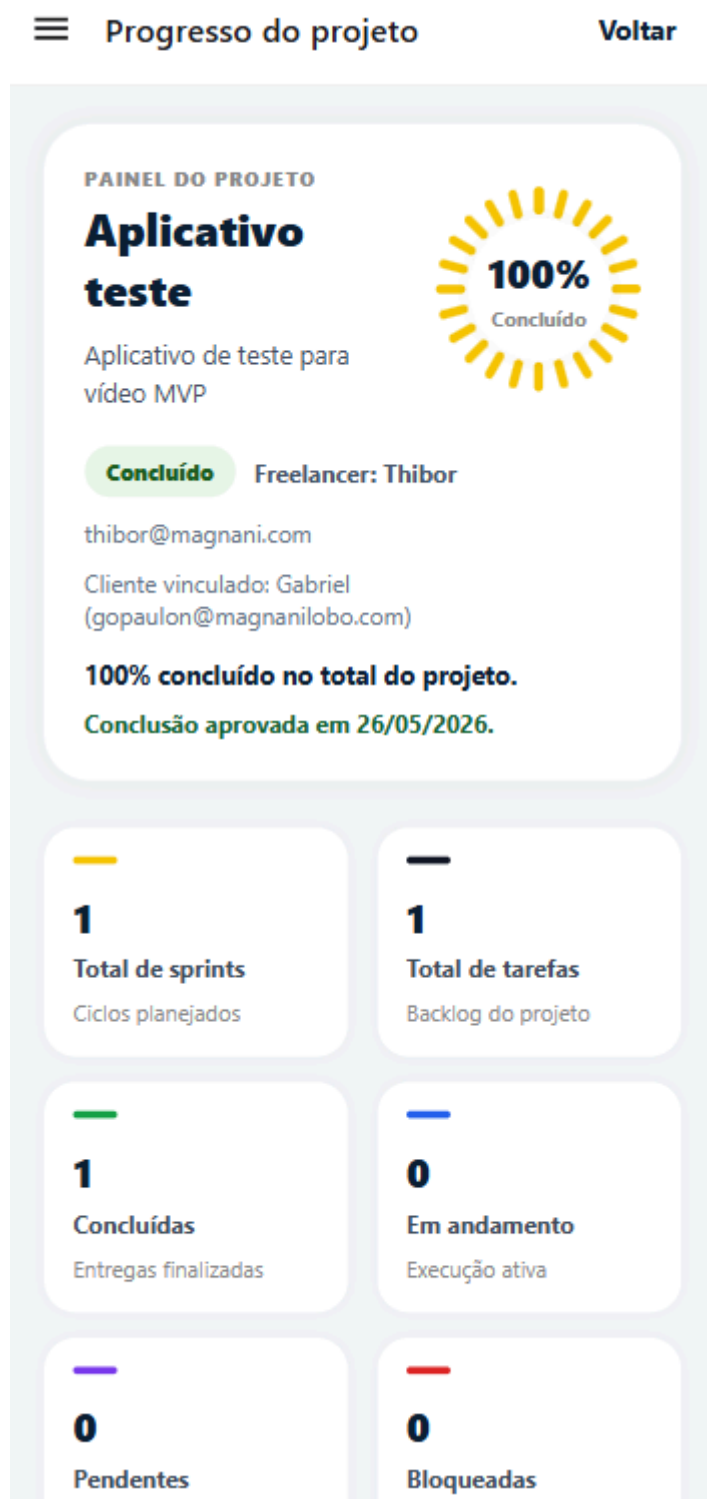
Figura 19 – Tela - Registrar tarefa concluída



Fonte: Autoria própria, 2026.

Na Figura 20 é apresentada a tela de visualização de progresso do projeto

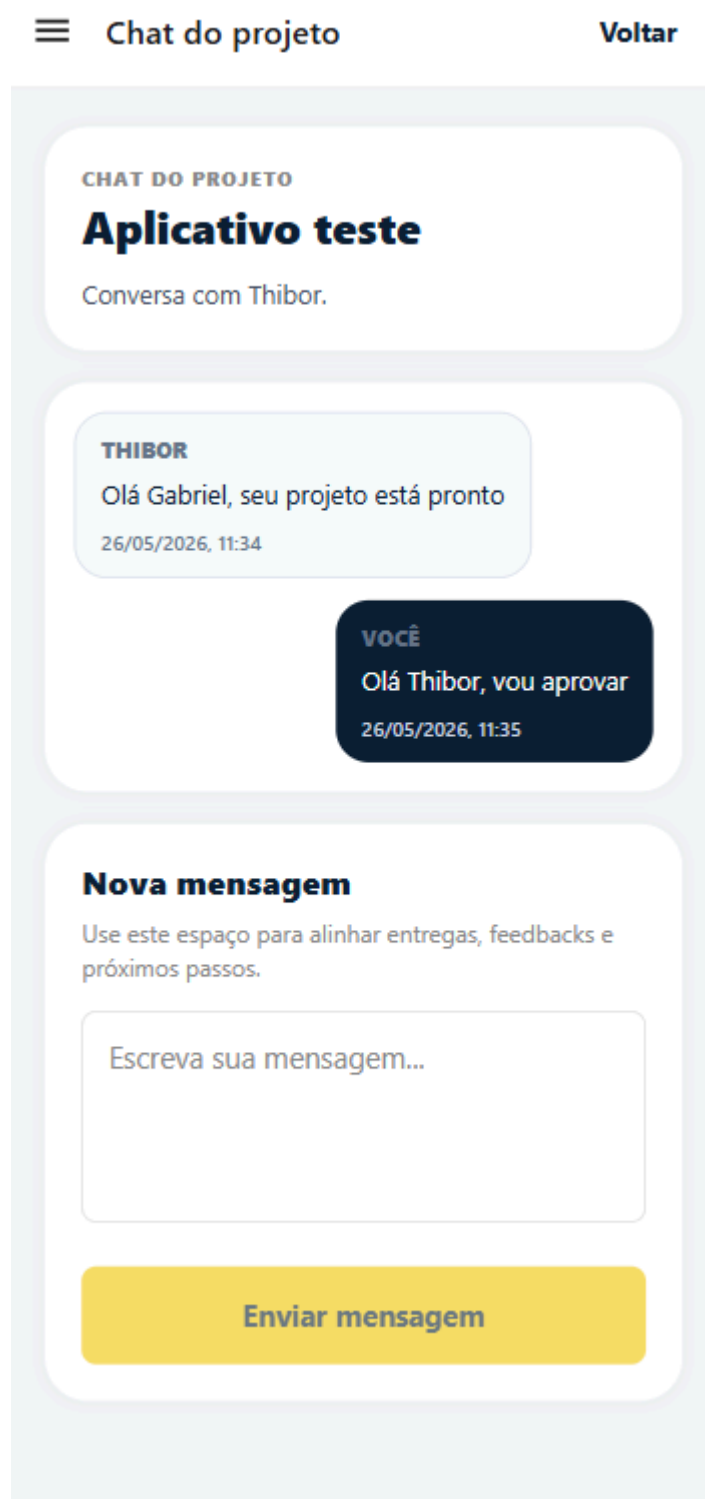
Figura 20 - Tela - Visualizar progresso do projeto



Fonte: Autoria própria, 2026.

Na Figura 21 é apresentada a tela de chat

Figura 21 - Tela - Chat do projeto



Fonte: Autoria própria, 2026.

Durante o desenvolvimento do projeto, foi utilizado o GitHub como plataforma de controle de versão e gerenciamento do código-fonte. A equipe optou pela separação do sistema em dois repositórios independentes, um destinado ao backend e outro ao frontend, permitindo melhor organização do projeto, isolamento entre módulos e maior facilidade na manutenção do código.

O uso do GitHub possibilitou o acompanhamento contínuo da evolução do sistema por meio do histórico de alterações, promovendo rastreabilidade, organização e colaboração entre os integrantes da equipe. Como prática recomendada para projetos colaborativos, destaca-se também a possibilidade de utilização do recurso GitHub Organizations, que permite centralizar repositórios, permissões e gerenciamento da equipe de forma mais estruturada.

Em relação ao versionamento, foi adotada uma estratégia de commits incrementais e padronizados, utilizando a convenção Conventional Commits, prática amplamente utilizada no desenvolvimento moderno para padronizar o histórico de alterações e facilitar a rastreabilidade do projeto.

Esse padrão permitiu identificar com clareza o tipo de alteração realizada em cada etapa do desenvolvimento, diferenciando implementações de novas funcionalidades, correções, documentação e tarefas de manutenção.

Entre os principais commits realizados durante o desenvolvimento do projeto, destacam-se:

- feat(front): inicializa app Expo com estrutura básica
- docs(readme): adiciona documentação do projeto
- feat: implementa CRUD de projeto e sprint
- feat: implementa fluxo de conclusão e aprovação de projeto
- feat: implementa autenticação de usuários
- feat: implementa gerenciamento de tarefas
- feat: implementa vinculação entre projetos e sprints
- fix(deps): corrige a instalação de pacotes expo
- fix: corrige validações e ajustes de fluxo
- chore(front): adiciona .env e atualiza .gitignore

A adoção desse padrão contribuiu para a organização do histórico de desenvolvimento, facilitando o acompanhamento da evolução do sistema, a identificação de alterações realizadas e a manutenção do projeto, além de demonstrar alinhamento com práticas profissionais amplamente utilizadas na engenharia de software.

Outra prática recomendada é a utilização de tags de versionamento, recurso do Git utilizado para marcar versões estáveis e entregas importantes do sistema, permitindo identificar pontos específicos da evolução do projeto.

A utilização de tags facilita a organização do histórico, recuperação de versões anteriores e controle formal das entregas realizadas ao longo do desenvolvimento.

Dessa forma, além da implementação técnica do sistema, o projeto incorporou práticas fundamentais de desenvolvimento colaborativo e controle de versão, demonstrando alinhamento com metodologias e padrões amplamente utilizados no mercado de desenvolvimento de software.

4 RESULTADOS

Esta seção apresenta os principais resultados alcançados com o desenvolvimento do trabalho, evidenciando como a solução atendeu aos objetivos propostos e quais foram os benefícios obtidos com sua implementação, além de propostas para trabalhos futuros.

4.1 Descrição dos resultados

O desenvolvimento do G Dev Flow resultou em uma plataforma funcional e integrada para o gerenciamento de projetos entre desenvolvedores freelancers e clientes, atendendo aos objetivos estabelecidos neste trabalho. A solução foi concebida para reduzir dificuldades relacionadas à organização das atividades, comunicação entre as partes e acompanhamento do progresso dos projetos, problemas frequentemente encontrados no contexto de desenvolvimento freelancer.

A aplicação disponibiliza recursos para cadastro e autenticação de usuários, criação e gerenciamento de projetos, organização do trabalho por meio de sprints e

tarefas, acompanhamento do progresso das atividades e comunicação entre desenvolvedor e cliente em um ambiente centralizado. Além disso, o sistema permite o compartilhamento de atualizações do projeto e a validação das entregas realizadas, proporcionando maior visibilidade sobre o andamento do desenvolvimento.

Durante a utilização da plataforma, observou-se que a centralização das informações e a estrutura baseada em metodologias ágeis contribuíram para uma melhor organização das demandas, maior controle sobre as entregas e maior transparência no relacionamento entre as partes envolvidas. A possibilidade de acompanhamento contínuo das atividades permitiu reduzir a dependência de comunicações externas e facilitar o alinhamento das expectativas entre cliente e desenvolvedor.

Dessa forma, os resultados obtidos demonstram que o G Dev Flow é capaz de oferecer uma solução eficiente para o gerenciamento de projetos freelancer, promovendo organização, rastreabilidade, transparência e comunicação em um único ambiente. Assim, a plataforma apresenta potencial para otimizar o fluxo de trabalho de profissionais autônomos, contribuindo para uma gestão mais eficiente e para o aumento da qualidade na execução dos projetos de software.

4.2 Perspectivas Futuras

Apesar dos resultados satisfatórios obtidos com o desenvolvimento da aplicação, existem diversas melhorias e extensões que podem ser implementadas em trabalhos futuros, visando ampliar as funcionalidades e aprimorar a experiência dos usuários como:

- implementação de notificações em tempo real, permitindo que usuários sejam informados imediatamente sobre atualizações em tarefas, comentários ou mudanças no projeto.
- Implementação de dashboards analíticos, com gráficos e métricas de desempenho, possibilitando uma visão mais detalhada do progresso dos projetos, produtividade e tempo de execução das tarefas.

- Implementação de um sistema de anexos, permitindo o compartilhamento de documentos, imagens e arquivos relacionados ao projeto diretamente na plataforma, centralizando as informações em um único ambiente.
- integração com ferramentas externas, como GitHub e GitLab, possibilitando sincronização de atividades, acompanhamento de commits e maior integração com o fluxo de desenvolvimento de software.
- Implementação de gestão financeira, a plataforma poderá oferecer recursos para controle de orçamento, registro de pagamentos e acompanhamento de valores acordados entre cliente e desenvolvedor.
- Adaptação da aplicação para ambiente web, além da versão mobile, pode ampliar seu alcance e facilitar o acesso por diferentes perfis de usuários, tornando a plataforma mais versátil e acessível.

Dessa forma, as propostas apresentadas representam oportunidades de expansão e aperfeiçoamento da plataforma, contribuindo para o aumento de sua funcionalidade, usabilidade e eficiência. A implementação dessas melhorias poderá agregar valor ao sistema e ampliar sua aplicabilidade no contexto do gerenciamento de projetos entre freelancers e clientes.

REFERÊNCIAS

ATLASSIAN. *What is Scrum?* 2023. Disponível em: <https://www.atlassian.com/agile/scrum>. Acesso em: 16 maio 2026.

EXPO. *Introduction to Expo*. 2024. Disponível em: <https://docs.expo.dev>. Acesso em: 3 maio 2026.

FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. 2000. Disponível em: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. Acesso em: 5 maio 2026.

HIBERNATE. *Hibernate ORM Documentation*. 2023. Disponível em: <https://hibernate.org/orm/documentation/>. Acesso em: 16 maio 2026.

IBM. *What is REST API?* 2023. Disponível em: <https://www.ibm.com/topics/rest-apis>. Acesso em: 23 abril 2026.

OBJECT MANAGEMENT GROUP. *OMG Unified Modeling Language (OMG UML)*, Version 2.5. Needham: Object Management Group, 2014.

SCHWABER, Ken; SUTHERLAND, Jeff. *Guia do Scrum*. 2020. Disponível em: <https://scrumguides.org>. Acesso em: 12 abril 2026.

SOMMERVILLE, Ian. *Engenharia de software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SPRING. *Spring Boot Documentation*. 2023. Disponível em: <https://spring.io/projects/spring-boot>. Acesso em: 2 maio 2026.