Sistema ACS

Sistema ACS | Docs

ACs UPE - Arquitetura

Descrição

Este artefato tem como objetivo documentar a arquitetura do sistema de atividades complementares da UPE (**ACs-UPE**). Ele proporciona uma visão abrangente da arquitetura do sistema, servindo como uma forma de comunicação entre os membros da equipe do projeto, abordando decisões arquiteturais significativas tomadas durante o desenvolvimento. O documento apresenta uma descrição detalhada da estrutura e organização do sistema, incluindo seus principais componentes, interfaces, fluxos de informação e dependências. Ele também destaca as principais decisões arquiteturais, como a escolha de tecnologias, padrões de projeto e estratégias de integração.

É essencial para garantir uma compreensão compartilhada entre os membros da equipe e permitir a colaboração efetiva no desenvolvimento do sistema. Além disso, serve como um ponto de referência para futuras discussões e análises de impacto, à medida que o projeto evolui e novas funcionalidades são adicionadas. Ao fornecer uma visão geral da arquitetura, o documento facilita a comunicação entre os stakeholders do projeto, incluindo desenvolvedores, gerentes de projeto e partes interessadas, permitindo que todos tenham uma compreensão clara das principais decisões arquiteturais e do funcionamento geral do sistema.

Em resumo, o objetivo deste artefato é documentar de forma clara e abrangente a arquitetura do sistema ACs-UPE, promovendo a comunicação efetiva entre os membros da equipe e permitindo uma base sólida para o desenvolvimento e evolução do projeto.

O que é o ACs-UPE?

O ACs-UPE é um sistema online que disponibiliza uma plataforma web para que os alunos possam enviar suas requisições de atividades complementares e acompanhar o processo de avaliação realizado pelos membros

avaliadores. Essa plataforma oferece uma maneira conveniente e acessível para os alunos registrarem suas atividades complementares, fornecendo uma interface interativa para o preenchimento dos dados e o envio das solicitações.

Uma vez submetidas, as requisições passam por um fluxo de avaliação, onde os membros avaliadores podem revisar e avaliar as atividades apresentadas pelos alunos. O sistema facilita esse processo, fornecendo ferramentas e recursos que permitem aos avaliadores realizar as avaliações de forma mais eficiente e precisa. Isso inclui a visualização dos detalhes da requisição, a análise dos documentos comprobatórios e a atribuição das horas correspondentes às atividades realizadas. Além disso, o ACs-UPE oferece recursos para o gerenciamento das requisições entre os avaliadores. Isso inclui a possibilidade de compartilhar informações e comentários sobre as requisições, facilitando a comunicação e colaboração entre os membros da equipe de avaliação. O sistema também fornece mecanismos para o controle do status das requisições, permitindo que os avaliadores definam se uma solicitação é deferida ou indeferida, garantindo uma gestão eficiente do fluxo de avaliação..

Tecnologias utilizadas

Front-end

- TypeScript
- React
- NextJs
- Styled Components

Back-end

- Java
- Spring boot

1. Requisitos

`/1. Requisitos`

Requisitos funcionais

RF 001	**Realizar cadastro via sistema**
Descrição	Um usuário deve ser capaz de realizar cadastro no sistema, vale ressaltar que para todo cadastro feito é atribuído ao usuário o perfil de aluno.
Atores	Nenhum.
Prioridade	Média.
Entrada e pré-condições	Entradas: Nome completo, periodo, telefone, email, senha, cep, rua, bairro, cidade, UF, numero; Pré-condições: Email institucional, senha com 8 ou mais caracteres incluindo caracteres especiais, letras maiúsculas e minúsculas.
Saída e pós-condições	Saídas: Confirmação de cadastro; Pós-condições: O usuário será direcionado para tela perfil para certificar que é membro da instituição após um processo de verificação.

RF 002	Realizar login via sistema
Descrição	Um usuário deve ser capaz de realizar login via sistema. Após a realização do login é retornado um JWT que identifica o usuário.
Atores	Usuário geral.
Prioridade	Média.
Entrada e pré-condições	Entradas: E-mail e senha; Pré-condições: O usuário deve estar cadastrado.
Saída e pós-condições	Saídas: Confirmação de login; Pós-condições: O usuário será direcionado para tela inicial.
RF 003	**Realizar login via google**
Descrição	O usuário deve ser capaz de fazer login no aplicativo utilizando o serviço de SSO (Single Sign-On) do Google, o OAuth2.
Atores	Nenhum.
Prioridade	Baixa.
Entrada e pré-condições	Entradas: Selecionar a conta do google; Pré-condições: Possuir uma conta google.
Saída e pós-condições	Saídas: Confirmação de login; Pós-condições: O usuário será direcionado para tela de perfil para completar os dados de cadastro.
RF 004	**Sair do sistema**
Descrição	O usuário deve ser capaz de sair do sistema.
Atores	Usuário geral.
Prioridade	Baixa.
Entrada e pré-condições	Entradas: Selecionar botão de logoff presente na tela inicial do sistema; Pré-condições: O usuário deve estar logado.
Saída e pós-condições	Saídas: Confirmação de logoff; Pós-condições: O usuário será direcionado para tela de login.
RF 005	**Verificar usuário institucional**
Descrição	O usuário deve realizar o processo de verificação, o qual receberá um token via e-mail para confirmar se é uma conta institucional própria.
Atores	Usuário geral.
Prioridade	Alta.
Entrada e pré-condições	Entradas: Inserir token recebido por e-mail; Pré-condições: O usuário não pode ser verificado e deve estar logado.
Saída e pós-	Saídas: Confirmação da verificação institucional do usuário; Pós-condições: O

condições	usuário será direcionado para tela inicial do sistema.
RF 006	**Alterar senha**
Descrição	O usuário deve ser capaz de alterar sua própria senha.
Atores	Usuário geral.
Prioridade	Baixa.
Entrada e pré-condições	Entradas: Inserir a senha antiga para ser possível realizar alteração da nova senha; Pré-condições: O usuário deve estar logado.
Saída e pós-condições	Saídas: Confirmação da alteração da senha; Pós-condições: O usuário recebe uma mensagem informando que a senha foi alterada com sucesso.
RF 007	**Recuperar senha**
Descrição	O usuário deve ser capaz de recuperar sua senha.
Atores	Usuário geral.
Prioridade	Baixa.
Entrada e pré-condições	Entradas: Inserir nova senha; Pré-condições: O usuário deve estar logado e recebe o link de recuperação de senha por e-mail.
Saída e pós-condições	Saídas: Confirmação da alteração da senha; Pós-condições: O usuário recebe uma mensagem informando que a senha foi alterada com sucesso.
RF 008	**Consultar dados do próprio usuário**
Descrição	O usuário deve ser capaz de visualizar todos os seus dados. Sendo possível visualizar o nome completo, cpf, período, telefone, e-mail, senha oculta, status de verificação, curso e endereço.
Atores	Usuário geral.
Prioridade	Baixa.
Entrada e pré-condições	Entradas: Não possui entradas; Pré-condições: O usuário deve estar logado.
Saída e pós-condições	Saídas: Não possuí saídas; Pós-condições: O usuário é direcionado para página de perfil.
RF 009	**Editar dados do próprio usuário**
Descrição	O usuário deve ser capaz de editar seus dados exceto e-mail, matrícula e CPF.
Atores	Usuário geral.
Prioridade	Baixa.
Entrada e pré-condições	Entradas: Inserir dados a serem alterados; Pré-condições: O usuário deve estar logado.

Saída e pós-condições	Saídas: Confirmação da alteração dos dados; Pós-condições: O usuário recebe uma mensagem informando que seus dados foram alterados com sucesso.

RF 010	**Solicitar apagar dados do usuário**
Descrição	O usuário deve ser capaz de solicitar que todos os seus dados sejam apagados do sistema.
Atores	Usuário geral.
Prioridade	Média.
Entrada e pré-condições	Entradas: Inserir senha do usuário para confirmação da exclusão; Pré-condições: O usuário deve estar logado.
Saída e pós-condições	Saídas: Confirmação da exclusão dos dados; Pós-condições: O usuário é redirecionado para a tela de login do sistema.

Perfil Aluno

RF 011	**Cadastrar requisição**
Descrição	O usuário deve ser capaz de cadastrar uma requisição com o intuito de ratificar uma determinada quantidade de horas das suas atividades complementares.
Atores	Aluno.
Prioridade	Alta.
Entrada e pré-condições	Entradas: É necessário inserir informações da requisição como o semestre e a quantidade de certificados. Para cada certificado é necessário informar título, descrição, data, horas e a atividade. Além dessas informações é necessário enviar os arquivos de certificados.o; Pré-condições: O usuário deve deve estar logado.
Saída e pós-condições	Saídas: Confirmação da envio da requisição; Pós-condições: O usuário é redirecionado para a tela específica da requisição enviada e a coordenação recebe a notificação por e-mail.

RF 012	**Consultar lista de requisições**
Descrição	O usuário deve ser capaz de visualizar sua lista de requisições.
Atores	Aluno.
Prioridade	Alta.
Entrada e pré-condições	Entradas: Não possui entradas; Pré-condições: O usuário deve estar logado.
Saída e pós-condições	Saídas: Não possuí saídas; Pós-condições: Caso selecione alguma requisição, o usuário é redirecionado para tela de requisição selecionada.

RF 013	**Buscar requisição específica**
Descrição	O usuário deve ser capaz de visualizar uma requisição em específico.

Atores	Aluno.
Prioridade	Média.
Entrada e pré-condições	Entradas: Não possui entradas; Pré-condições: O usuário deve estar logado.
Saída e pós-condições	Saídas: Não possuí saídas; Pós-condições: O usuário é redirecionado para tela de requisição selecionada.
RF 014	**Receber alerta após mudança de status de uma requisição**
Descrição	O usuário deve receber um alerta via e-mail após haver uma mudança no status da requisição.
Atores	Aluno.
Prioridade	Alta.
Entrada e pré-condições	Entradas: Não possui entradas; Pré-condições: Quando o status de uma requisição associada a um determinado aluno for alterada.
Saída e pós-condições	Saídas: Não possuí saídas; Pós-condições: O usuário recebe um e-mail com as informações daquela requisição e o seu respectivo status.
RF 015	**Baixar arquivo PDF referente à requisição**
Descrição	O usuário deve ter a opção de fazer o download da requisição em formato PDF.
Atores	Aluno.
Prioridade	Baixa.
Entrada e pré-condições	Entradas: Não possui entradas; Pré-condições: O usuário deve estar logado e selecionar a opção de fazer download do arquivo em PDF.
Saída e pós-condições	Saídas: Confirmação do download do arquivo; Pós-condições: O usuário vê o arquivo baixado na pasta selecionada.
RF 016	**Visualizar indicadores sobre as requisições enviadas**
Descrição	Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.
Atores	Usuário com perfil de aluno.
Prioridade	Média.
Entrada e pré-condições	Entradas: Em construção; Pré-condições: Em construção.
Saída e pós-condições	Saídas: Em construção; Pós-condições: Em construção.

Perfil Coordenador

RF 017	Solicitar permissões de coordenação
Descrição	O usuário pode solicitar ao administrador o perfil de coordenação, é necessário validar se o usuário é docente e faz parte da coordenação de um determinado curso.
Atores	Usuário geral.
Prioridade	Média.
Entrada e pré-condições	Entradas: O usuário deve enviar alguma documentação que confirme a veracidade da sua posição; Pré-condições: O usuário deve estar logado no sistema.
Saída e pós-condições	Saídas: Confirmação da solicitação de autorização; Pós-condições: O usuário deve receber por e-mail um retorno sobre a sua solicitação.

RF 018	Consultar lista de requisições
Descrição	O coordenador deve ser capaz de visualizar a listas de requisições encaminhadas para a coordenação e comissão do seu respectivo curso.Também deve ser possível filtrar as requisições por campos de interesse, como: data, semestre, status da requisição e aluno.
Atores	Coordenador.
Prioridade	Média.
Entrada e pré-condições	Entradas: Nenhuma; Pré-condições: O usuário deve estar logado.
Saída e pós-condições	Saídas: Não possuí saídas; Pós-condições: Caso selecione alguma requisição, o usuário é redirecionado para tela de requisição selecionada.

RF 019	Encaminhar requisição para a comissão
Descrição	O coordenador deve ser capaz de encaminhar a requisição para a comissão do seu curso.
Atores	Coordenador.
Prioridade	Alta.
Entrada e pré-condições	Entradas: Não possui entradas; Pré-condições: O usuário deve estar logado.
Saída e pós-condições	Saídas: Confirmação de encaminhamento de requisição; Pós-condições: O coordenador recebe uma mensagem informando que a requisição foi encaminhada com sucesso e a comissão recebe por e-mail a notificação;

RF 020	Assinar documento requisição
Descrição	Antes de deferir ou indeferir uma requisição, o coordenador deve ser capaz de baixar o documento da requisição na sua máquina e submetê-lo assinado ao sistema.
Atores	Coordenador.
Prioridade	Alta.

Entrada e pré-condições	Entradas: O usuário deve submeter o documento assinado; Pré-condições: O usuário deve estar logado e selecionar a opção de fazer download do arquivo em PDF, assiná-lo e realizar a submissão.
Saída e pós-condições	Saídas: Confirmação de submissão do documento assinado; Pós-condições: O coordenador recebe uma mensagem informando que a requisição foi assinada com sucesso;
RF 021	**Ratificar requisição**
Descrição	O coordenador deve ser capaz de finalizar o fluxo da requisição, deferindo-a ou indeferindo-a.
Atores	Coordenador.
Prioridade	Alta.
Entrada e pré-condições	Entradas: Nenhuma; Pré-condições: O usuário deve estar logado e em casos de aprovação a requisição deve ter passado pela comissão.
Saída e pós-condições	Saídas: Confirmação da ratificação da requisição; Pós-condições: O coordenador recebe uma mensagem informando que a requisição foi ratificada e encaminhada para a escolaridade;
RF 022	**Atribuir perfil para a comissão**
Descrição	O coordenador pode atribuir perfis de comissão para os usuários da sua escolha.
Atores	Coordenador.
Prioridade	Média.
Entrada e pré-condições	Entradas: Informar o usuário que deseja adicionar as permissões de comissão; Pré-condições: O usuário deve estar logado no sistema.
Saída e pós-condições	Saídas: Confirmação da atribuição do perfil; Pós-condições: O usuário que recebeu as permissões deve receber por e-mail a notificação.
RF 023	**Visualizar indicadores sobre as requisições enviadas**
Descrição	Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
Atores	Coordenador.
Prioridade	Média.
Entrada e pré-condições	Entradas: Em construção; Pré-condições: Em construção.

Saída e pós-condições	Saídas: Em construção; Pós-condições: Em construção.

Comissão

RF 024	**Consultar lista de requisições**
Descrição	A comissão deve ser capaz de visualizar a listas de requisições encaminhadas para a comissão do seu respectivo curso. Também deve ser possível filtrar as requisições por campos de interesse, como: data, semestre e aluno.
Atores	Comissão.
Prioridade	Média.
Entrada e pré-condições	Entradas: Nenhuma; Pré-condições: O usuário deve estar logado.
Saída e pós-condições	Saídas: Não possuí saídas; Pós-condições: Caso selecione alguma requisição, o usuário é redirecionado para tela de requisição selecionada.
RF 025	**Adicionar quantidade de horas nas atividades complementares**
Descrição	A comissão deve ser capaz de adicionar a quantidade de horas por certificados de uma requisição.
Atores	Comissão.
Prioridade	Alta.
Entrada e pré-condições	Entradas: Inserir quantidade de horas para o certificado; Pré-condições: O usuário deve estar logado.
Saída e pós-condições	Saídas: Confirmação da inserção das horas dos certificados; Pós-condições: O usuário retorna para a tela da requisição.
RF 026	**Validar requisição**
Descrição	A comissão deve ser capaz de finalizar o fluxo da requisição, deferindo-a ou indeferindo-a. Após isso é submetido para a coordenação realizar a validação final.
Atores	Comissão.
Prioridade	Alta.
Entrada e pré-condições	Entradas: Nenhuma; Pré-condições: O usuário deve estar logado e em casos de aprovação a requisição deve ser enviada para a coordenação.
Saída e pós-condições	Saídas: Confirmação da ratificação da requisição; Pós-condições: O coordenador recebe um e-mail informando que a requisição foi finalizada pela comissão;

Requisitos não funcionais

Desempenho

Esta área define as métricas de desempenho que deverão ser atingidas para que o sistema tenha uma boa usabilidade e não afete a experiência do usuário.

RNF 001	**Tempo de Resposta**
Descrição:	O aplicativo deverá apresentar tempos de resposta inferiores a 1000 ms, de forma a apresentar um carregamento suficientemente rápido.
Prioridade:	Alta

RNF 002	**Usuários online**
Descrição:	O aplicativo deve ser altamente eficiente e capaz de gerenciar perfeitamente a conexão de até 100 usuários online simultaneamente, proporcionando uma experiência fluida e sem interrupções para todos os usuários.
Prioridade:	Alta

Disponibilidade

RNF 003	**Período ativo**
Descrição:	O aplicativo deve permanecer online 24 horas por dia, 7 dias por semana, com tolerância a interrupções de no máximo 2 horas em situações excepcionais para manutenções ou atualizações planejadas.
Prioridade:	Média

Hardware

RNF 004	**Compatibilidade**
Descrição:	Para o perfeito funcionamento do sistema, é necessário apenas possuir um navegador com acesso à internet. Essa simplicidade na exigência de recursos garante que o aplicativo seja facilmente acessível e utilizado por uma ampla variedade de usuários.
Prioridade:	Alta

Segurança

RNF 005	**Criptografia**
Descrição:	O sistema deve priorizar a segurança dos dados e a proteção da privacidade dos usuários, adotando o uso do protocolo HTTPS (Hyper Text Transfer Protocol Secure) como uma camada de criptografia confiável. Ao utilizar o HTTPS, todas as informações transmitidas entre o aplicativo e os usuários serão criptografadas, garantindo a confidencialidade e a integridade dos dados durante a comunicação.
Prioridade:	Baixa

RNF 006	**Autenticação**

Descrição:	O usuário deve ser capaz de realizar login através de sua conta Google utilizando o serviço SSO (Single Sign-On) da Google além de poder fazer pelo próprio sistema utilizando a lógica de autenticação com Spring security e JWT.
Prioridade:	Alta

Documentação

RNF 007	**Documentação APIs REST**
Descrição:	É fundamental que o endpoint do aplicativo seja cuidadosamente documentado, a fim de facilitar integrações futuras com outros sistemas e permitir correções eficientes de implementações no front-end. Nesse sentido, o uso da ferramenta Swagger desempenha um papel crucial. O Swagger proporciona uma abordagem estruturada e padronizada para documentação de API, permitindo que os desenvolvedores compreendam facilmente a funcionalidade e os parâmetros do endpoint.
Prioridade:	Alta

1.1 Requisitos do Sistema

`/1. Requisitos/1.1 Requisitos do Sistema`

1.2 Requisitos Arquiteturais

`/1. Requisitos/1.2 Requisitos Arquiteturais`

2. Níveis

`/2. Níveis`

O modelo C4 é uma abordagem altamente amigável e de fácil aprendizado para a diagramação de arquitetura de software. Ao criar diagramas de arquitetura de software bem elaborados, é possível desempenhar um papel fundamental na comunicação eficaz, tanto internamente, entre as equipes de desenvolvimento e produto de software, quanto externamente, com outros stakeholders.

Esses diagramas são uma ferramenta valiosa para auxiliar na integração eficiente de novos membros da equipe, pois fornecem uma visão clara e concisa da estrutura e interações do sistema. Além disso, eles possibilitam análises e avaliações detalhadas da arquitetura, permitindo identificar pontos fortes e fracos, bem como oportunidades de melhoria.

Outro benefício importante dos diagramas de arquitetura de software é a capacidade de identificar riscos potenciais, como a tempestade de riscos, através da visualização clara das dependências e interações entre os componentes do sistema. Isso possibilita a tomada de medidas preventivas e o planejamento de estratégias para mitigar os riscos identificados.

Além disso, os diagramas de arquitetura também são úteis na modelagem de ameaças, permitindo visualizar possíveis vulnerabilidades e definir medidas de segurança adequadas. Com um entendimento aprofundado da arquitetura do sistema, é possível tomar decisões informadas para garantir a proteção adequada dos dados e a segurança geral do sistema.

Nível 1: Diagrama de contexto do sistema

Um diagrama de contexto do sistema é um excelente ponto de partida para a diagramação e documentação de um sistema de software, pois permite uma visão geral do mesmo. Nele, é possível representar o sistema como uma caixa central, cercada por seus usuários e outros sistemas com os quais interage. Desenhar esse tipo de diagrama proporciona uma compreensão mais clara e abrangente do sistema, permitindo uma análise mais completa de sua interação com o ambiente ao seu redor.

O detalhe não é importante aqui, pois esta é a sua visão ampliada mostrando uma imagem grande da paisagem do sistema. O foco deve estar nas pessoas (atores, funções, personas, etc.) e sistemas de software, em vez de tecnologias, protocolos e outros detalhes de baixo nível. É o tipo de diagrama que você pode mostrar para pessoas não técnicas.

Nível 2: Diagrama de contêiner

Após obter uma compreensão de como seu sistema se encaixa no ambiente geral de TI, uma etapa extremamente útil é expandir os limites do sistema por meio de um diagrama de contêiner. Um "contêiner" pode ser entendido como um componente como um aplicativo da Web do lado do servidor, um aplicativo de página única, um aplicativo de desktop, um aplicativo móvel, um esquema de banco de dados, um sistema de arquivos, entre outros. Basicamente, um contêiner é uma unidade autônoma e implantável separadamente, como um espaço de processo isolado, capaz de executar código ou armazenar dados de forma independente.

O diagrama do contêiner mostra a forma de alto nível da arquitetura de software e como as responsabilidades são distribuídas por ela. Ele também mostra as principais opções de tecnologia e como os contêineres se comunicam entre si. É um diagrama simples e focado em tecnologia de alto nível que é útil para desenvolvedores de software e equipe de suporte/operações.

Nível 3: Diagrama de componentes

É possível expandir e decompor cada contêiner de forma mais detalhada, a fim de identificar os principais blocos de construção estruturais e suas interações. O diagrama de componentes oferece uma visão clara de como um contêiner é composto por diversos "componentes", fornecendo informações sobre a natureza de cada um desses componentes, suas responsabilidades e os detalhes relacionados à tecnologia e implementação utilizada. Esse tipo de diagrama permite uma compreensão mais aprofundada da estrutura do sistema, destacando os elementos fundamentais que o compõem e como eles se relacionam entre si.

Nível 4: Diagrama de código

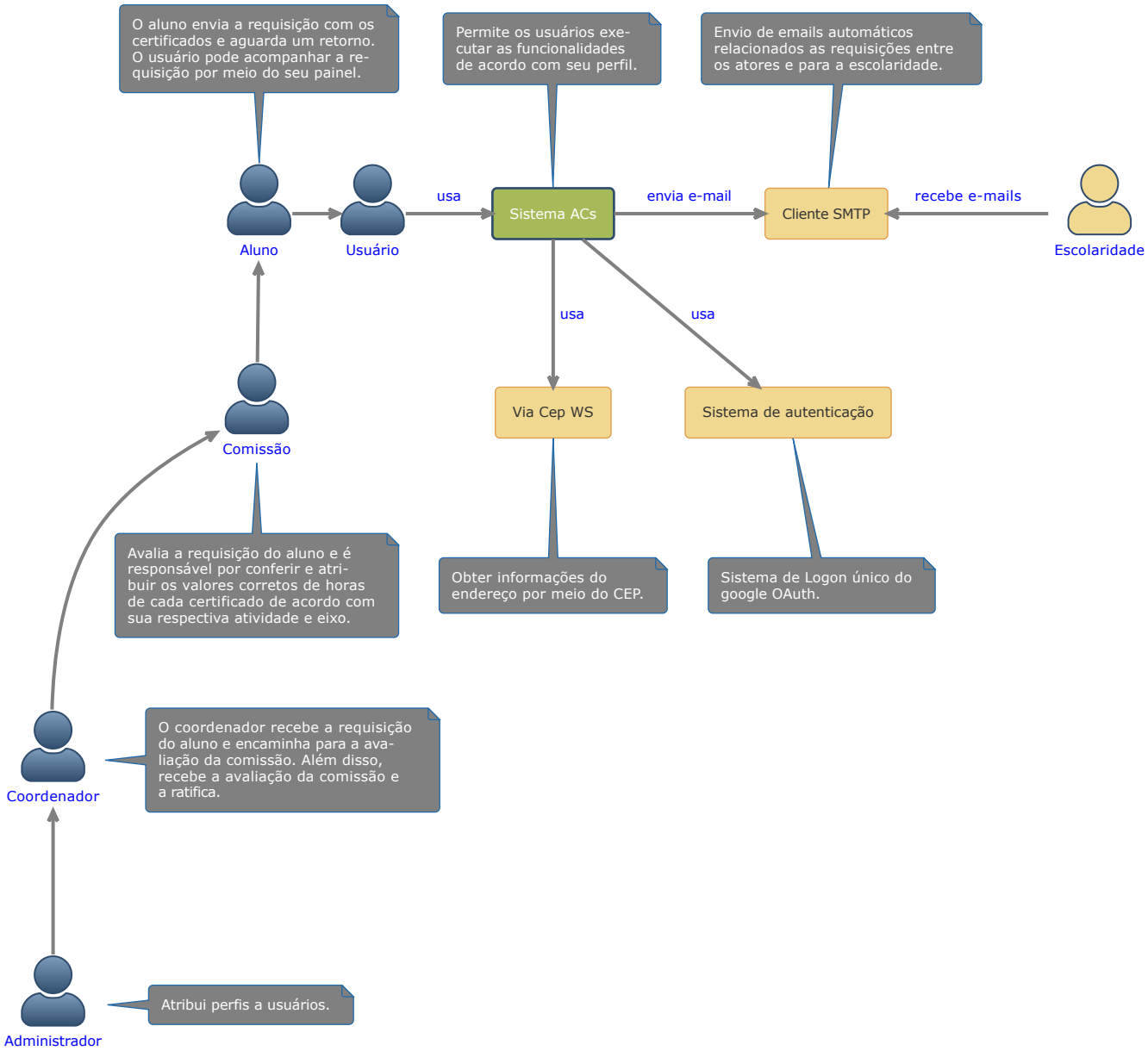
Por fim, é possível aprofundar cada componente para exibir como ele é implementado em forma de código, utilizando diagramas de classe UML, diagramas de entidade-relacionamento ou técnicas similares. Esses diagramas fornecem uma representação visual das relações entre as classes, entidades ou componentes, destacando seus atributos, métodos e associações. Ao utilizar tais diagramas, você pode comunicar de forma eficaz os detalhes técnicos da implementação do sistema e evidenciar o funcionamento interno de cada componente.

Este é um nível de detalhe opcional e, normalmente, pode ser acessado sob demanda por meio de ferramentas como IDEs. Idealmente, esse tipo de diagrama deveria ser gerado automaticamente utilizando ferramentas como uma IDE de modelagem ou UML. Ao criar esses diagramas, é importante considerar mostrar apenas os atributos e métodos relevantes que ajudem a contar a história desejada. É importante ressaltar que esse nível de

detalhe não é recomendado para todos os componentes, sendo mais adequado para destacar os componentes mais importantes ou complexos.

2.1 Contexto - C1

`/2. Níveis/2.1 Contexto - C1`



O principal objetivo do diagrama de contexto, de acordo com o modelo C4, é fornecer uma visão geral e simplificada da arquitetura do sistema, destacando seu relacionamento com o ambiente externo. O diagrama de contexto mostra o sistema em foco como uma única caixa central, cercada por elementos externos, como usuários, sistemas externos e outras entidades com as quais o sistema interage.Esse diagrama é útil para comunicar de forma clara e concisa a função e o propósito do sistema, bem como sua interação com o ambiente ao seu redor. Ele fornece uma representação de alto nível da arquitetura, permitindo que os stakeholders identifiquem facilmente as partes interessadas envolvidas e entendam a amplitude do sistema.

Além disso, o diagrama de contexto é uma base sólida para a elaboração de outros diagramas mais detalhados, pois fornece um ponto de partida para a decomposição do sistema em componentes menores. Ele ajuda a

definir os limites e a identificar as interfaces do sistema, sendo uma ferramenta essencial para a compreensão inicial da arquitetura em um nível mais amplo.

- **Escopo:** Um único sistema de software.
- **Elementos primários:** O sistema de software no escopo.
- **Elementos de suporte:** Pessoas (por exemplo, usuários, atores, funções ou personas) e sistemas de software (dependências externas) que estão diretamente conectados ao sistema de software no escopo.
- **Público alvo:** Todos, técnicos e não técnicos, dentro e fora da equipe de desenvolvimento de software.

Principais elementos

Personas

Cada usuário possui obrigatoriamente um perfil, que define os níveis de autorização no sistema.

- **Aluno:** Esse usuário representa aqueles que possuem acesso ao sistema para enviar requisições juntamente com suas atividades complementares. É importante ressaltar que apenas os membros da Universidade de Pernambuco possuem permissão para acessar o sistema.
- **Comissão:** Esse usuário desempenha um papel fundamental na avaliação das requisições de atividades complementares. Sua responsabilidade principal é verificar cuidadosamente os dados da requisição e os certificados apresentados. Além disso, é sua responsabilidade atribuir a quantidade de horas correspondente às atividades e definir o status da requisição como deferido ou indeferido, com base nas informações fornecidas e nos critérios estabelecidos.
- **Coordenação:** O responsável pelo gerenciamento das requisições possui o perfil de coordenador. Ele é o primeiro a receber as requisições e realiza validações iniciais, tendo autoridade para indeferir uma requisição sem envolvê-la com a comissão responsável. Somente um usuário com esse perfil possui a autoridade para concluir o fluxo da requisição. O coordenador desempenha um papel crucial na triagem inicial, garantindo que apenas as requisições válidas e adequadas prossigam para o próximo estágio do processo.
- **Administrador:** Esse usuário é responsável por atribuir os perfis de comissão e coordenação. Para realizar essa atribuição, é necessário enviar um e-mail para a conta de administração, solicitando o cadastro de usuários com esses perfis específicos. Esse processo garante que apenas usuários autorizados possuam os privilégios necessários para desempenhar as funções de comissão e coordenação, mantendo assim a segurança e a organização do sistema.

Personas Externas

- **Escolaridade:** Esse ator não possui interação direta com o sistema, porém desempenha um papel crucial ao receber e-mails durante o processo de cadastro das requisições. Ele é notificado quando um aluno realiza o cadastro de uma requisição e também quando a requisição é concluída. Sua responsabilidade é inserir as informações das atividades complementares no Siga, garantindo que os registros sejam atualizados adequadamente. Embora não esteja diretamente envolvido no sistema, esse ator desempenha uma função indispensável para o fluxo de trabalho eficiente e preciso.

Sistema principal

- **Sistema ACs-UPE:** Essa plataforma web é um sistema que oferece aos alunos a capacidade de cadastrar requisições, ao mesmo tempo em que permite o gerenciamento dessas requisições pela

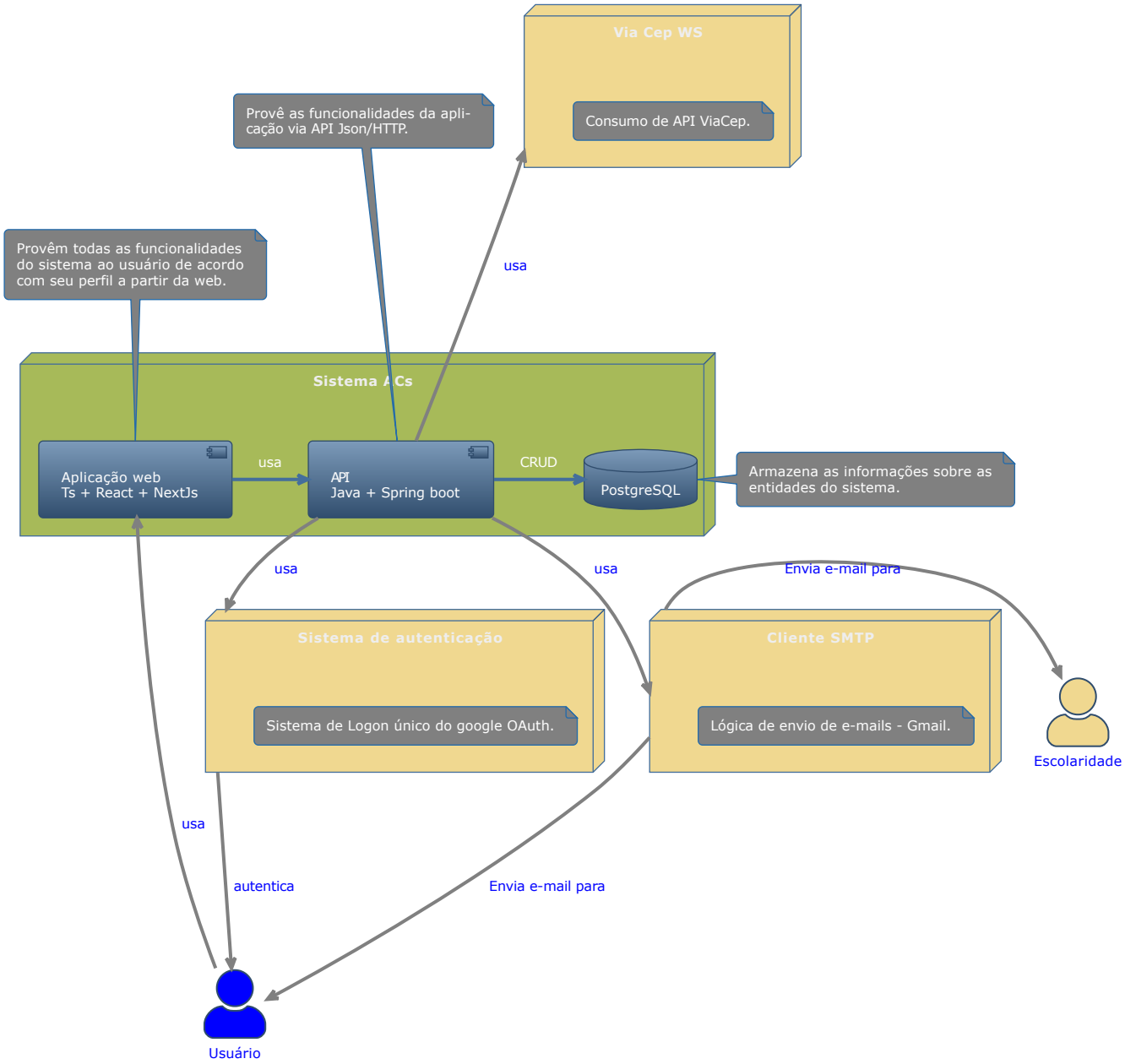
coordenação e comissão. É um ambiente centralizado que facilita o processo de solicitações de atividades complementares, promovendo uma comunicação eficiente entre os alunos e as equipes responsáveis pela avaliação e aprovação das requisições. A plataforma oferece recursos que permitem aos alunos inserir os dados necessários para as requisições, enquanto a coordenação e a comissão podem acessar e administrar essas requisições de forma organizada, efetiva e segura.

Sistema externos

- **Sistema de e-mail:** Responsável pelo envio de e-mails automáticos relacionados às requisições para a escolaridade, envio de e-mails de notificação entre os atores do sistema, bem como e-mails relacionados às etapas de configuração de contas de usuário. Esse componente desempenha um papel crucial na comunicação efetiva do sistema, garantindo que as partes relevantes sejam informadas sobre o status das requisições, atualizações e outras informações importantes. Os e-mails automáticos são disparados de acordo com eventos específicos, agilizando o fluxo de trabalho e mantendo os usuários informados em tempo hábil.
- **Sistema de autenticação:** O recurso de login com credenciais do Google oferece aos usuários a conveniência de utilizar suas contas do Google para acessar diversos serviços e aplicativos. Isso elimina a necessidade de criar novas credenciais e permite um processo de login simplificado e seguro. Ao aproveitar as credenciais existentes do Google, os usuários podem acessar rapidamente os serviços desejados, reduzindo o atrito no processo de autenticação e melhorando a experiência do usuário. Além disso, o login com credenciais do Google também proporciona um nível adicional de confiança e segurança, uma vez que a autenticação é realizada pela infraestrutura robusta e confiável do Google.
- **API do Via CEP:** O recurso permite obter automaticamente os dados de endereço do usuário apenas informando o CEP. Com essa funcionalidade, é possível simplificar e agilizar o processo de preenchimento de informações de endereço, eliminando a necessidade de digitar manualmente cada campo. Ao inserir o CEP, o sistema realiza uma busca em uma base de dados atualizada e recupera automaticamente as informações de rua, número, bairro, cidade e estado associados ao CEP fornecido. Isso proporciona uma experiência mais conveniente para o usuário, reduzindo erros de digitação e economizando tempo ao preencher formulários ou realizar transações online que requerem informações de endereço.

2.2 Container - C2

`/2. Níveis/2.2 Container – C2`



Após compreender como o sistema de Atividades Complementares (ACs) se encaixa no ambiente geral de Tecnologia da Informação (TI), a próxima etapa consiste em ampliar os limites do sistema por meio de um diagrama de contêiner. Um "contêiner" pode ser entendido como uma entidade autônoma, como um aplicativo da web do lado do servidor, um aplicativo de página única, um aplicativo desktop, um aplicativo móvel, um esquema de banco de dados, um sistema de arquivos ou uma API, entre outros exemplos. Em essência, um contêiner é uma unidade independente e implantável, que opera em seu próprio espaço de processo, executando código ou armazenando dados.

Ao representar esses contêineres em um diagrama, é possível visualizar a estrutura e as interações entre eles. Isso proporciona uma visão clara da arquitetura do sistema, identificando os componentes principais e suas dependências. O diagrama de contêiner ajuda a compreender como essas entidades autônomas colaboram para fornecer funcionalidades do sistema de ACs e como se relacionam com outros sistemas e recursos de TI. Esse diagrama é uma ferramenta valiosa para comunicar a arquitetura do sistema, permitindo que as equipes de desenvolvimento, gerentes de projeto e outras partes interessadas tenham uma visão abrangente da estrutura do sistema e das tecnologias envolvidas. Ele serve como base para a análise, aprimoramento e manutenção contínua da arquitetura, facilitando o entendimento do sistema e auxiliando na tomada de decisões técnicas.

- **Escopo:** Um único sistema de software.
- **Elementos primários:** Contêineres dentro do sistema de software no escopo.
- **Público alvo:** Técnicos dentro e fora da equipe de desenvolvimento de software, incluindo arquitetos de software, desenvolvedores e equipe de operações/suporte.
- **Observações:** Este diagrama não diz nada sobre cenários de implantação, clustering, replicação e failover etc.

Principais elementos

Containers

- **Aplicação web:**
    - A plataforma web foi desenvolvida utilizando o framework React e Next.js, juntamente com a linguagem TypeScript. Essa combinação tecnológica proporciona uma base sólida para a criação de uma interface interativa e responsiva, que permite aos usuários interagir com todas as funcionalidades do sistema.
    - Ao utilizar o framework React, a plataforma se beneficia de uma arquitetura de componentes reutilizáveis, o que facilita a criação e a manutenção de uma interface consistente e escalável. O Next.js oferece recursos avançados, como renderização do lado do servidor e roteamento simplificado, que aprimoram o desempenho e a experiência do usuário.
    - A integração com a API é essencial para garantir a funcionalidade completa da plataforma. Por meio de comunicação eficiente com a API, a interface pode enviar e receber dados de forma transparente, permitindo que os usuários realizem ações e obtenham informações atualizadas em tempo real.
    - Essa abordagem de desenvolvimento, utilizando React, Next.js e TypeScript, resulta em uma plataforma web moderna, robusta e escalável, que oferece aos usuários uma experiência interativa e completa. A combinação dessas tecnologias possibilita a criação de interfaces de usuário atraentes e funcionais, ao mesmo tempo em que facilita a integração eficiente com os dados do sistema.
- **API:**
    - A API de backend desempenha um papel central no sistema, sendo desenvolvida com as tecnologias Java e Spring Boot. Essa combinação tecnológica é altamente adequada para lidar com o processamento de solicitações dos usuários, gerenciar a lógica de negócios e facilitar a integração com bancos de dados, sistemas externos e serviços.
    - Ao utilizar Java e Spring Boot, a API se beneficia de recursos avançados e padrões de desenvolvimento estabelecidos. O Java é uma linguagem amplamente adotada, conhecida por sua estabilidade, desempenho e suporte à orientação a objetos. O Spring Boot, por sua vez, é um framework que simplifica o desenvolvimento de aplicativos Java, oferecendo recursos integrados, configuração automática e uma arquitetura baseada em convenções.
    - Essa escolha tecnológica permite que a API seja robusta, escalável e segura. O Java e o Spring Boot fornecem um ambiente confiável para o processamento de solicitações, garantindo que o sistema possa lidar com um grande volume de tráfego e oferecer desempenho consistente. Além disso, o ecossistema do Spring Boot oferece recursos de segurança abrangentes, como autenticação e autorização, que podem ser facilmente implementados.
    - Por meio dessa combinação de Java e Spring Boot, a API de backend se torna uma solução eficiente e confiável para o sistema, capaz de atender às necessidades de processamento de dados, gerenciamento de lógica de negócios e integração com outros componentes do sistema.

2.3 Componente - C3

`/2. Níveis/2.3 Componente - C3`

O objetivo principal dos diagramas de componentes no C4 Model é mostrar a estrutura interna dos contêineres do sistema e como eles são compostos por diferentes componentes. Os diagramas de componentes fornecem uma visão detalhada dos principais blocos de construção estruturais do sistema e suas interações. Esses diagramas ajudam a entender a arquitetura de um sistema de software em um nível mais granular, identificando os componentes individuais e suas responsabilidades. Eles mostram como os componentes se relacionam entre si, quais são as dependências e como eles colaboram para fornecer as funcionalidades do sistema.

- **Escopo:** Um único contêiner.
- **Elementos primários:** Componentes dentro do contêiner no escopo.
- **Público alvo:** Arquitetos e desenvolvedores de software.

Convenções Back-end

Antes de nos aprofundarmos no C3, é importante ressaltar que são utilizadas algumas convenções que apoiam a arquitetura proposta. Essas convenções são as seguintes

- Convenção de nomenclaturas de arquivos e diretórios (Oracle)
- Convenção de fluxo de trabalho Git - com adapatações e aderência parcial (Gitflow)
- Convenção de commits (Conventional Commits)

Convenções Front-end

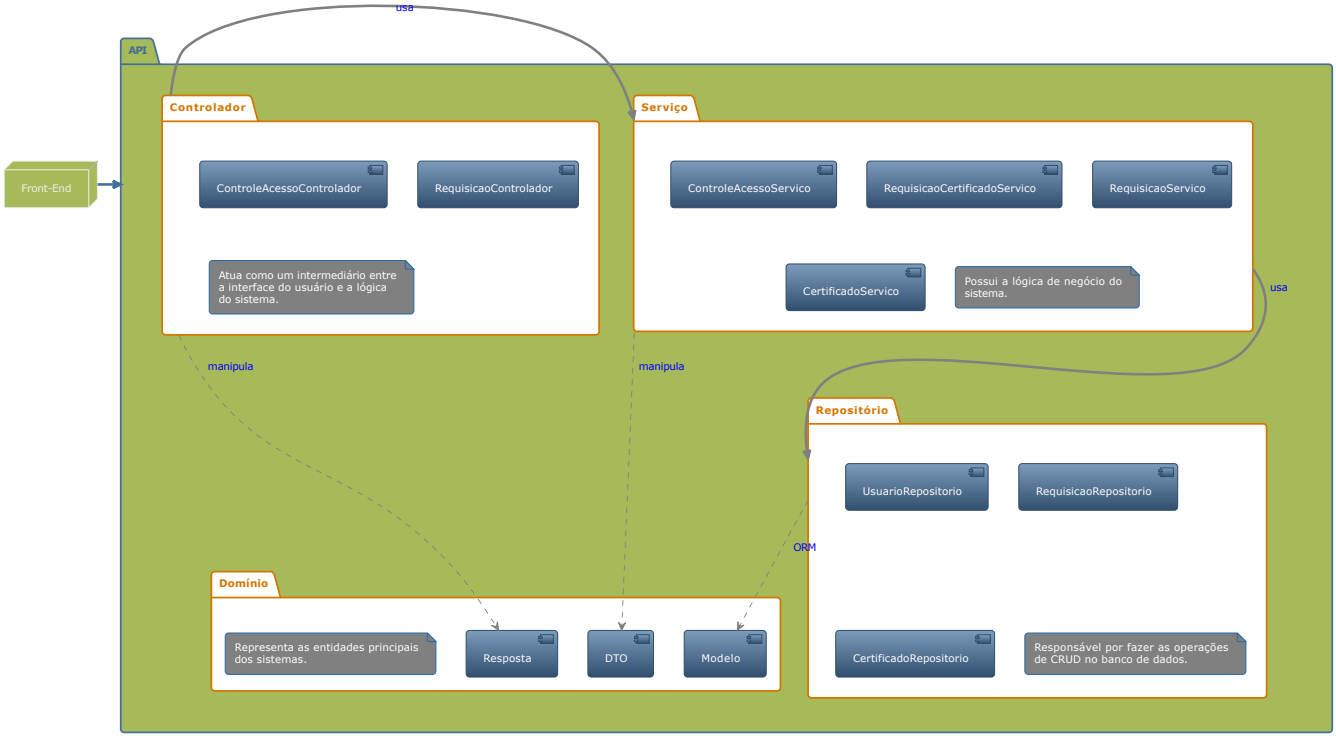
⚒🏗

Restrições e metas arquiteturais

- As tecnologias de desenvolvimento utilizadas são:
    - Front-end: TypeScript associado aos frameworks React + NextJs.
    - Back-end: Java associado ao framework Spring boot.
- O projeto não possui nenhuma espécie de orçamento, tornando a aquisição de serviços de terceiros um custo pessoal, portanto os serviços utilizados devem ser gratuitos e o servidor pessoal ou da instituição;.
- A aplicação será disponibilizada como página web.
- O sistema deve permanecer online 24 horas por dia, 7 dias por semana, com tolerância a interrupções de no máximo 2 horas em situações excepcionais para manutenções ou atualizações planejadas.
- O serviço de API deve ser criado com um banco de dados completo, com gerenciamento de concorrência e maior capacidade de armazenamento, dado que o sistema será desenvolvido com foco em manter múltiplos usuários acessando concorrentemente.

2.3.1-Back

`/2. Níveis/2.3 Componente - C3/2.3.1-Back`



O diagrama de componentes em questão oferece uma representação visual dos componentes principais do sistema. Esses componentes são elementos fundamentais que desempenham papéis específicos no funcionamento do sistema como um todo.

Back-end

Convenções

Para promover a separação de responsabilidades, a coesão e a manutenção da arquitetura, são adotados padrões de nomenclatura para pastas, arquivos, funções e variáveis. As pastas são organizadas para representar as diferentes camadas do software e são nomeadas como "controlador", "serviço", "repositório" e "modelo". Os arquivos seguem a convenção de "nome da entidade + nome da camada", por exemplo, "RequisicaoController". Em relação às funções e variáveis, é recomendado utilizar o camelCase, pois isso melhora a legibilidade do código.

Camadas

- **Controlador:** O Controlador atua como uma ponte entre a interface do usuário e a lógica de negócio da aplicação. Ele recebe os dados fornecidos pelo usuário, faz a validação dos parâmetros de entrada e decide qual ação precisa ser tomada com base nas informações recebidas. Essa camada também é responsável por traduzir as respostas do serviço em uma representação adequada para a interface do usuário, como um JSON ou uma página HTML.
- **Serviço:** A camada Serviço, por sua vez, contém a lógica de negócio da aplicação. Ela é responsável por processar as requisições recebidas do Controller, realizar as operações necessárias e coordenar as interações entre diferentes componentes do sistema. O serviço encapsula as regras de negócio e pode fazer chamadas a outras camadas, como a camada de acesso a dados (Repositório), para buscar ou persistir informações no banco de dados.
- **Repositório:** O Repositório é a camada responsável pelo acesso aos dados. Ele fornece métodos para recuperar, armazenar e manipular informações no banco de dados ou em outros meios de

armazenamento. Essa camada abstrai os detalhes do acesso ao banco de dados, permitindo que o serviço trabalhe com objetos de domínio sem precisar conhecer os detalhes da implementação do banco de dados.

- **Modelo:** A camada Modelo representa os objetos de domínio da aplicação. Ela define as entidades e seus atributos, bem como os relacionamentos entre elas. Os objetos de domínio são usados pelo serviço e pelo Repositório para manipular as informações da aplicação de acordo com as regras de negócio.
- **Config:** Inclui várias classes de configuração, como as responsáveis pela autenticação e autorização, além das classes de configuração do Swagger.
- **Utils:** Possui os utilitários do sistema (Classes de exceção).

Novos casos de uso

Para implementar novos casos de uso na API, siga o passo-a-passo abaixo:

- Crie o **modelo** para a entidade em questão, definindo seus atributos.
- Crie o **repositório** responsável pela camada de acesso aos dados da entidade.
- Crie o **serviço** que irá implementar a lógica de negócio relacionada à entidade.
- Crie o **controlador** responsável por receber as requisições relacionadas à entidade e retornar as respostas correspondentes.

2.4 Código - C4

`/2. Níveis/2.4 Código - C4`

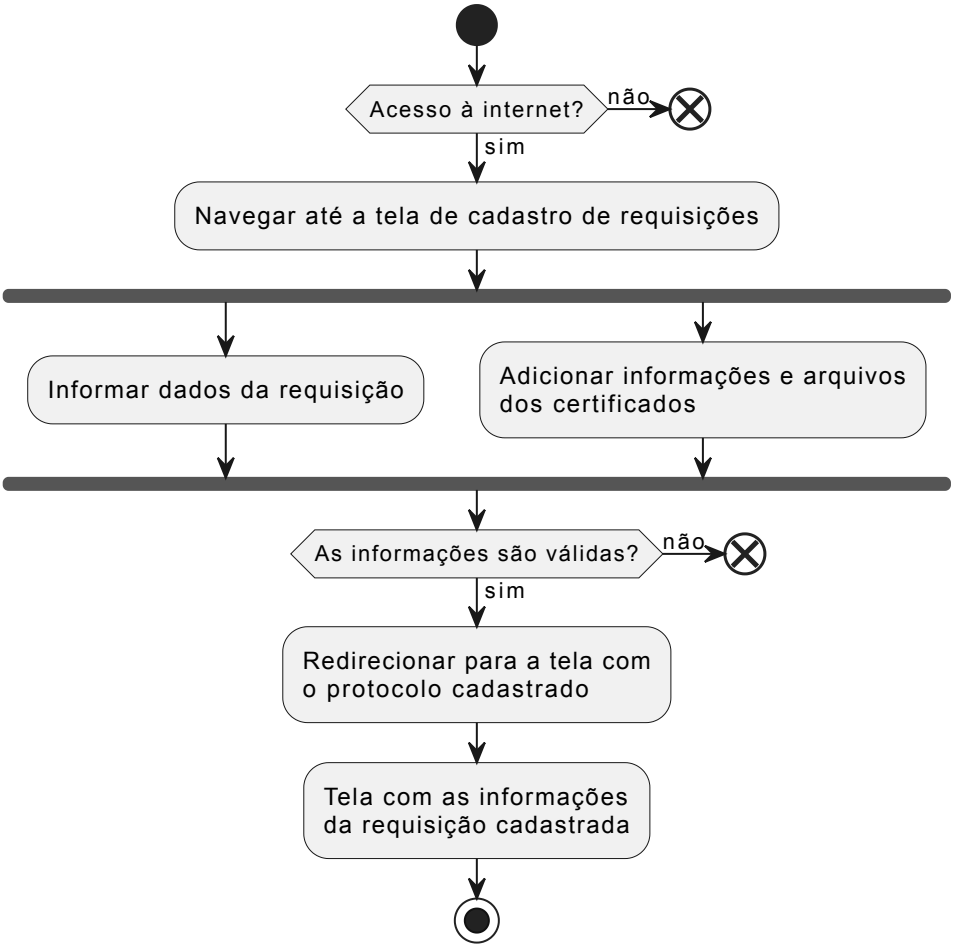
O nível de código no C4 Model tem como objetivo principal representar a implementação detalhada dos componentes de software em um sistema. Ela está relacionada à perspectiva de implementação técnica do sistema, mostrando como o código é organizado, estruturado e interage entre si. Permite aos desenvolvedores e engenheiros de software entenderem a estrutura interna dos componentes, como classes, métodos, interfaces e relacionamentos, além de possibilitar a análise das dependências entre eles.

Este nível fornece informações cruciais sobre como o sistema está sendo implementado em termos de código. Ela ajuda a garantir a consistência e a clareza na implementação, permitindo uma compreensão detalhada da estrutura e do funcionamento interno dos componentes de software. Ao visualizar a camada de código, os desenvolvedores podem identificar padrões de design, resolver problemas de implementação, entender a lógica de negócio e realizar melhorias no código existente. Além disso, ela pode servir como uma referência valiosa para a manutenção, depuração e evolução do sistema ao longo do tempo.

- **Escopo:** Um único componente.
- **Elementos primários:** Elementos de código (por exemplo, classes, interfaces, objetos, funções, tabelas de banco de dados, etc.) dentro do componente no escopo.
- **Elementos de suporte:** Containers (dentro do sistema de software em escopo) mais pessoas e sistemas de software diretamente conectados aos componentes.
- **Público alvo:** Arquitetos e desenvolvedores de software.

2.4.1-Atividades

`/2. Níveis/2.4 Código - C4/2.4.1-Atividades`

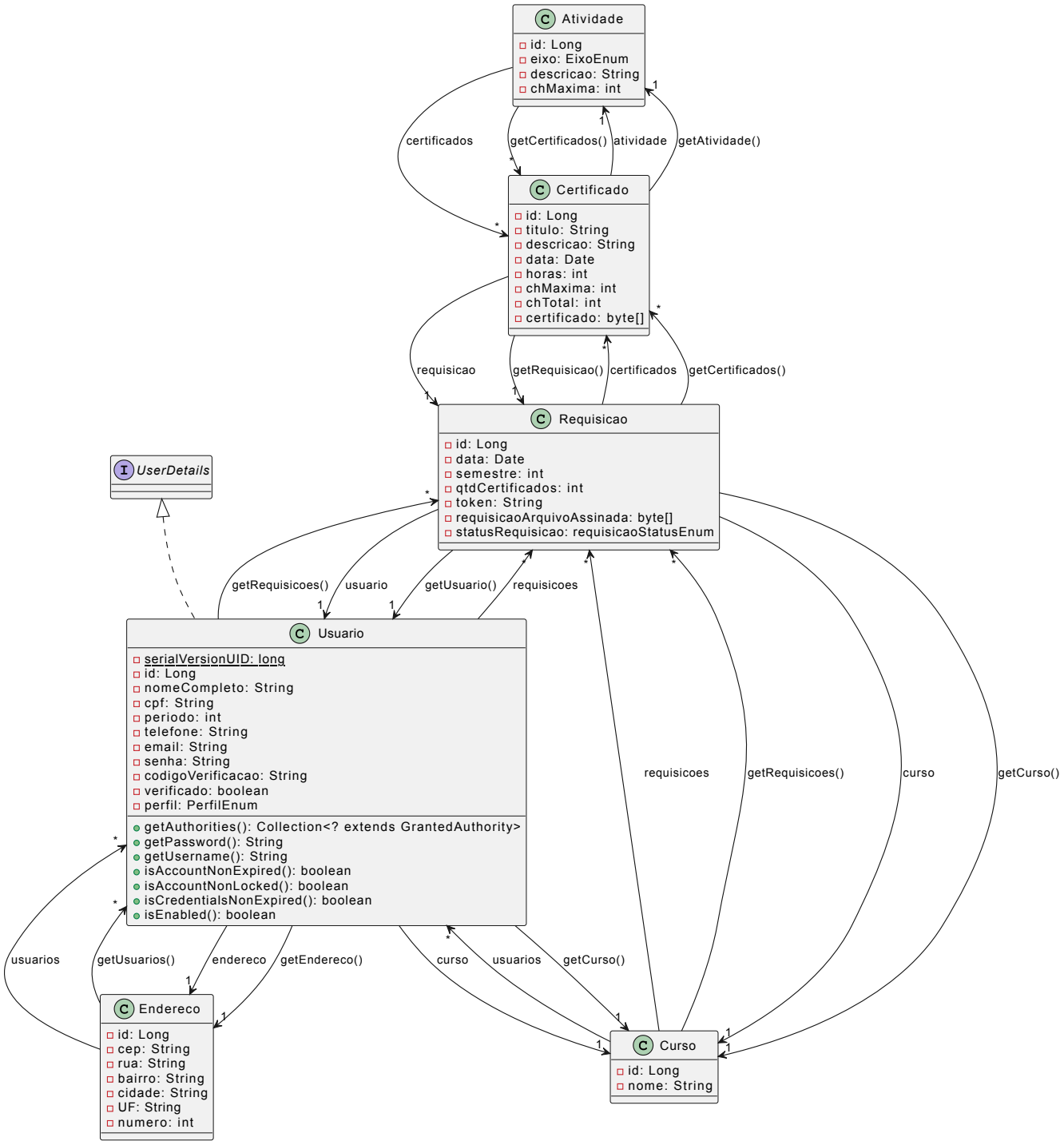


A imagem acima denota o fluxo de cadastro de uma requisição na perspectiva do usuário. O diagrama de atividades UML é uma representação gráfica que permite descrever o fluxo de controle e as atividades executadas em um sistema. É uma ferramenta valiosa para modelar processos de negócio, fluxos de trabalho e comportamentos complexos. No contexto específico do sistema em questão, a principal atividade é o cadastro da requisição. Para ilustrar o fluxo de atividades que devem ser realizadas tanto pelo usuário quanto pelo aplicativo para alcançar os casos de uso desejados, foram criados diagramas de atividades específicos. Esses diagramas de atividades fornecem uma visualização clara e concisa das etapas e ações necessárias para realizar as funcionalidades do sistema. Eles ajudam a compreender o sequenciamento das atividades, as condições de guarda e as interações entre o usuário e o aplicativo.

É importante ressaltar que, para executar essas funcionalidades, é necessário ter acesso à internet, conforme ilustrado nos diagramas. Essa observação destaca a dependência do sistema em relação à conectividade para garantir o pleno funcionamento das atividades. Em resumo, os diagramas de atividades UML são usados para descrever o fluxo de controle e as atividades em um sistema, sendo especialmente úteis para modelar processos de negócio e fluxos de trabalho complexos. No contexto específico do sistema em questão, os diagramas de atividades foram criados para representar o fluxo de atividades necessárias para alcançar os casos de uso.

2.4.2-Classes

`/2. Níveis/2.4 Código - C4/2.4.2-Classes`



A imagem acima denota as principais entidades que compõem o sistema e os seus relacionamentos. O diagrama de classes UML é uma representação visual que ilustra a estrutura estática de um sistema, exibindo as classes, seus atributos, métodos, relacionamentos e associações. Ele desempenha um papel fundamental na modelagem orientada a objetos, permitindo uma visualização clara e organizada da organização e composição das classes em um sistema de software. Além de fornecer uma visão estrutural, o diagrama de classes pode ser expandido para mostrar a implementação de cada componente por meio de diagramas como UML (Linguagem de Modelagem Unificada), diagramas de entidade-relacionamento ou outros similares. No entanto, é importante destacar que esse nível de detalhe é opcional e geralmente disponível mediante solicitação em ferramentas como IDEs (Ambientes de Desenvolvimento Integrado).

Idealmente, os diagramas detalhados seriam gerados automaticamente por meio de ferramentas especializadas, como uma ferramenta de modelagem UML ou IDE, simplificando o processo de criação e atualização dos diagramas. Ao criar esses diagramas, é recomendável mostrar apenas os atributos e métodos relevantes para

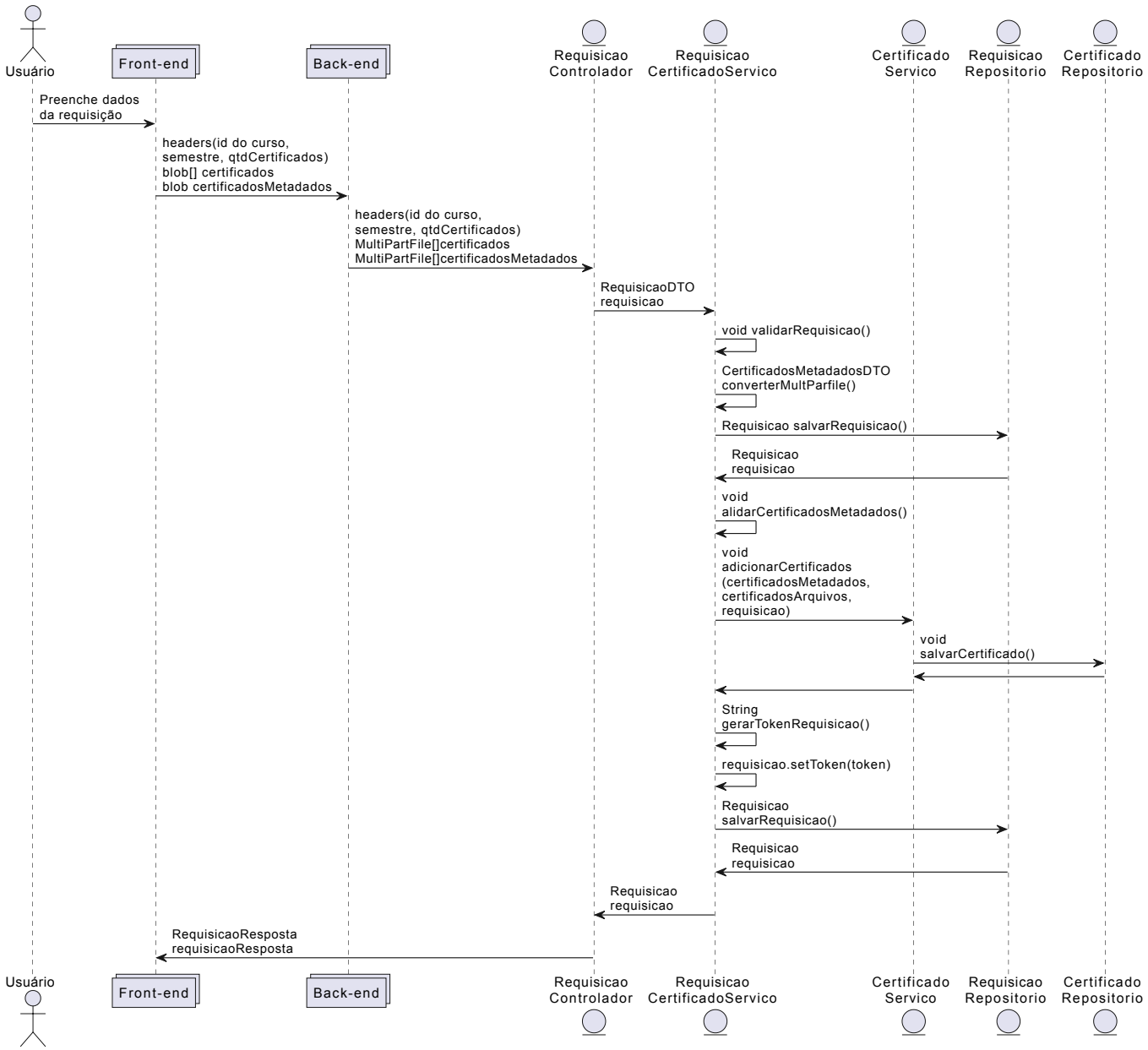
transmitir a história ou a narrativa desejada, evitando informações excessivas que possam dificultar a compreensão.

Portanto, é aconselhável aplicar esse nível de detalhe apenas aos componentes mais importantes ou complexos, a fim de manter a clareza e a concisão do diagrama de classes. Essa abordagem ajuda a comunicar efetivamente a estrutura do sistema, facilitando o trabalho de desenvolvedores, arquitetos e outras partes interessadas na compreensão do sistema de software em questão.

- **Escopo:** Um único componente.
- **Elementos primários:** Elementos de código (por exemplo, classes, interfaces, objetos, funções, tabelas de banco de dados, etc.) dentro do componente no escopo.
- **Elementos de suporte:** Containers (dentro do sistema de software em escopo) mais pessoas e sistemas de software diretamente conectados aos componentes.
- **Público Alvo:** Arquitetos e desenvolvedores de software.

2.4.3-Sequência

`/2. Níveis/2.4 Código - C4/2.4.3-Sequência`



A imagem acima denota o diagrama de sequência relacionado ao de cadastro de uma requisição, sendo um dos principais fluxos do sistema. Os diagramas de sequência oferecem uma representação gráfica do comportamento de uma funcionalidade, levando em consideração a interação entre todos os componentes do software envolvidos em seu uso. Isso inclui a chamada de funções e seus retornos, bem como a participação de atores, controllers e services.

Ao analisar um diagrama de sequência, é possível identificar de forma clara e visual como o sistema se comporta e qual é o fluxo de chamadas de cada função. Essa representação fornece um suporte valioso na compreensão do funcionamento interno do sistema, destacando as interações entre os componentes e como eles colaboram para o cumprimento dos requisitos do caso de uso em questão.Os diagramas de sequência facilitam a identificação de possíveis problemas ou oportunidades de melhoria. Essa abordagem ajuda a garantir que o sistema esteja sendo projetado e implementado de acordo com os requisitos e expectativas dos usuários, permitindo uma análise detalhada do fluxo de execução e das interações entre os diferentes elementos do sistema.

Portanto, ao utilizar os diagramas de sequência, é possível obter uma compreensão mais clara e precisa do comportamento do sistema, auxiliando no processo de desenvolvimento e no aprimoramento contínuo do software.