

FACULDADE DE INFORMÁTICA E ADMINISTRAÇÃO PAULISTA  
FIAP

GABRIEL GENARO RM551986  
PALOMA MIRELA RM551321  
VICTOR KENZO RM551649  
RICARDO RAMOS RM550166  
ARTHUR BALDISSERA RM550219



## SUMÁRIO

Sumário.....	2
Descrição do Projeto .....	3
Análise de requisitos .....	4
Requisitos Funcionais .....	4
Regras de Negócio.....	5
Design do sistema.....	6
Diagrama de componentes .....	6
Diagrama de Entidades(Banco de Dados).....	7
API Endpoints .....	8



## DESCRIÇÃO DO PROJETO

A Hamm API (nomeada carinhosamente em homenagem ao porquinho-cofrinho do filme *Toy Story*) é uma aplicação desenvolvida em ASP.NET Core Web API com Entity Framework Core, cujo principal objetivo é atuar como o motor de um sistema de gestão financeira pessoal.

Ela é responsável por centralizar e disponibilizar, de forma segura e organizada, todas as informações relacionadas a:

- Gastos pessoais
- Metas financeiras
- Budgets mensais
- Relatórios consolidados

A API funciona como o ponto central de comunicação entre um aplicativo cliente (por exemplo, um app mobile) e o banco de dados, garantindo que todas as operações de cadastro, consulta, atualização e exclusão de dados sejam realizadas de forma consistente.

Além disso, a Hamm API implementa a lógica de negócios necessária para:

- Processar e integrar dados financeiros.
- Gerar relatórios (JSON).
- Executar consultas avançadas utilizando LINQ.
- Integrar informações externas, como taxas de câmbio em tempo real, por meio de APIs públicas.

Com isso, o projeto busca oferecer uma solução modular e extensível, capaz de evoluir futuramente para um sistema completo de gestão financeira pessoal com interface mobile ou web, mantendo a API como núcleo central.



# ANÁLISE DE REQUISITOS

## Requisitos Funcionais

### RF01 – Gerenciar Usuários (CRUD Usuários)

- A API deve permitir cadastrar, editar, listar e excluir usuários.
- A API deve armazenar dados básicos do usuário (nome, e-mail, senha).

### RF02 – Gerenciar Transações (CRUD Transações)

- A API deve permitir o cadastro, edição, listagem e exclusão de transações financeiras (entrada e saída).
- Cada transação deve estar associada a um usuário e conter: valor, data, categoria, descrição, tipo (receita ou despesa) e forma de pagamento.

### RF03 – Gerenciar Categorias de Despesas/Receitas

- A API deve permitir cadastrar e consultar categorias (ex.: alimentação, transporte, lazer, contas, investimentos, metas, etc).
- As categorias devem ser utilizadas para classificar transações.

### RF04 – Gerenciar Orçamentos Mensais (Budgets)

- A API deve permitir definir um orçamento mensal por categoria de despesa.
- A API deve permitir consultar quanto do orçamento já foi utilizado e quanto ainda está disponível para cada categoria e no total.

### RF05 – Gerenciar Metas Financeiras

- A API deve permitir cadastrar, editar, listar e excluir metas financeiras (ex.: juntar R\$ 5.000 para uma viagem).
- A API deve calcular o progresso das metas com base nas contribuições feitas.

### RF06 – Relatórios Financeiros

- A API deve permitir gerar relatórios consolidados dos dados de determinado mês ou determinado ano em .json.
- A API deve gerar estatísticas, como:
  - Total de receitas e despesas no mês ou ano fornecido.
  - Orçamento total do período (somando todos os orçamentos definidos para aquele período).
  - Top 5 maiores despesas e Top 5 maiores receitas (com base no Tipo das transações).
  - Gasto por categoria.
  - Status e detalhes de todas as metas com data alvo futuro



## RF08 – Consultar Taxas de Câmbio (Integração Externa)

- A API deve integrar-se com uma API externa de câmbio para obter taxas de conversão de moedas (API escolhida: <https://br.dolarapi.com/v1/cotacoes>).
- O usuário deve poder consultar quanto ele poderia comprar de certa moeda com determinado valor em reais (as moedas disponíveis são: ARS, USD, EUR, CLP, UYU)

## Regras de Negócio

### RN01 – Usuários

- Cada usuário deve ter **e-mail único**; não é permitido cadastrar dois usuários com o mesmo e-mail.
- Senhas devem ser armazenadas de forma **criptografada**.
- Um usuário só pode acessar e manipular suas próprias transações, categorias, budgets e metas.

### RN02 – Transações

- Transações **não podem ter valor negativo** para receitas ou despesas; apenas o tipo define se é entrada ou saída.
- A data da transação não pode ser futura (não é possível registrar transações em datas que ainda não ocorreram).
- Cada transação deve estar vinculada a uma **categoria existente**.
- O tipo de pagamento (dinheiro, cartão, transferência) deve ser consistente e validado.

### RN03 – Categorias

- Categorias padrão (alimentação, transporte, lazer, contas, investimentos) já podem vir pré-cadastradas na API.
- Usuários podem criar categorias, mas não podem alterar ou excluir categorias padrão do sistema.

### RN04 – Orçamentos Mensais (Budgets)

- O orçamento definido para uma categoria **não pode ser negativo**.

### RN06 – Relatórios

- Relatórios gerados devem refletir apenas as transações **pertencentes ao usuário** que solicitou.
- Estatísticas (top 5 despesas, saldo consolidado, gasto por categoria) devem ser calculadas **em tempo real** com base nos dados atuais do banco.



#### RN08 – Exportação de Arquivos

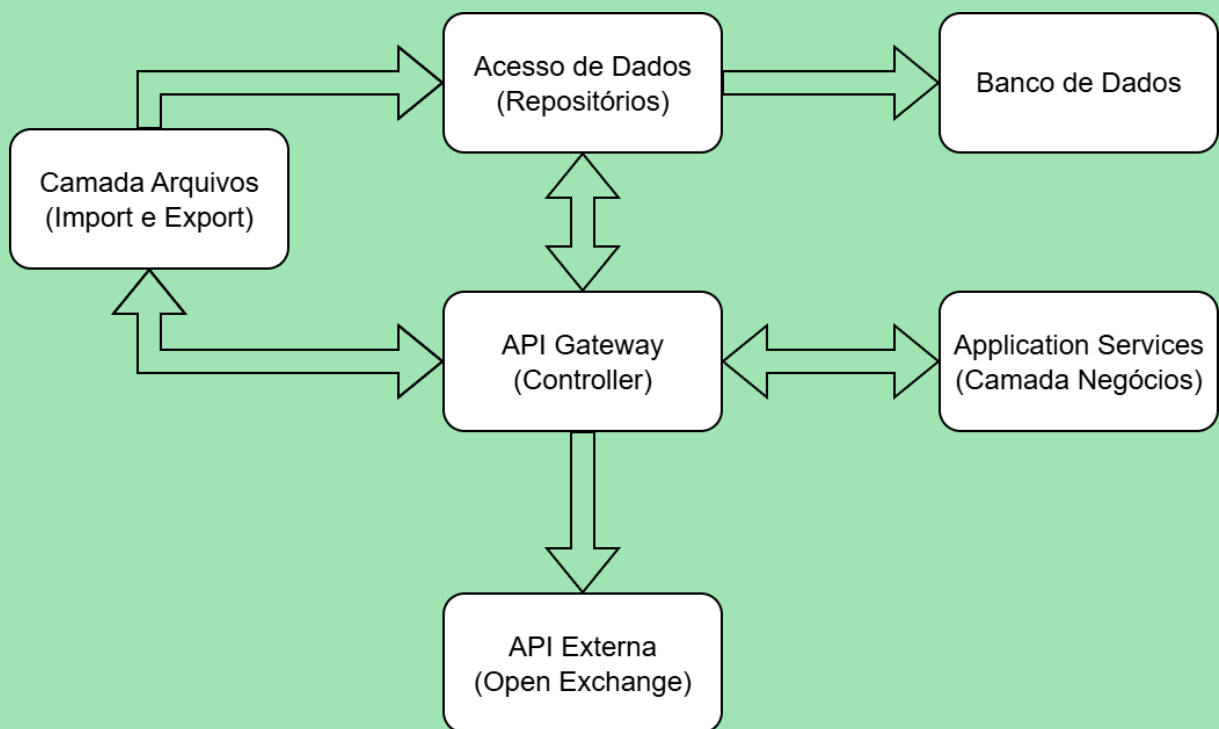
- Usuário pode escolher período ou categorias para exportação.
- Arquivos exportados devem incluir **todas as informações relevantes**: valor, data, categoria, tipo, forma de pagamento.

#### RN09 – Consultar Taxas de Câmbio

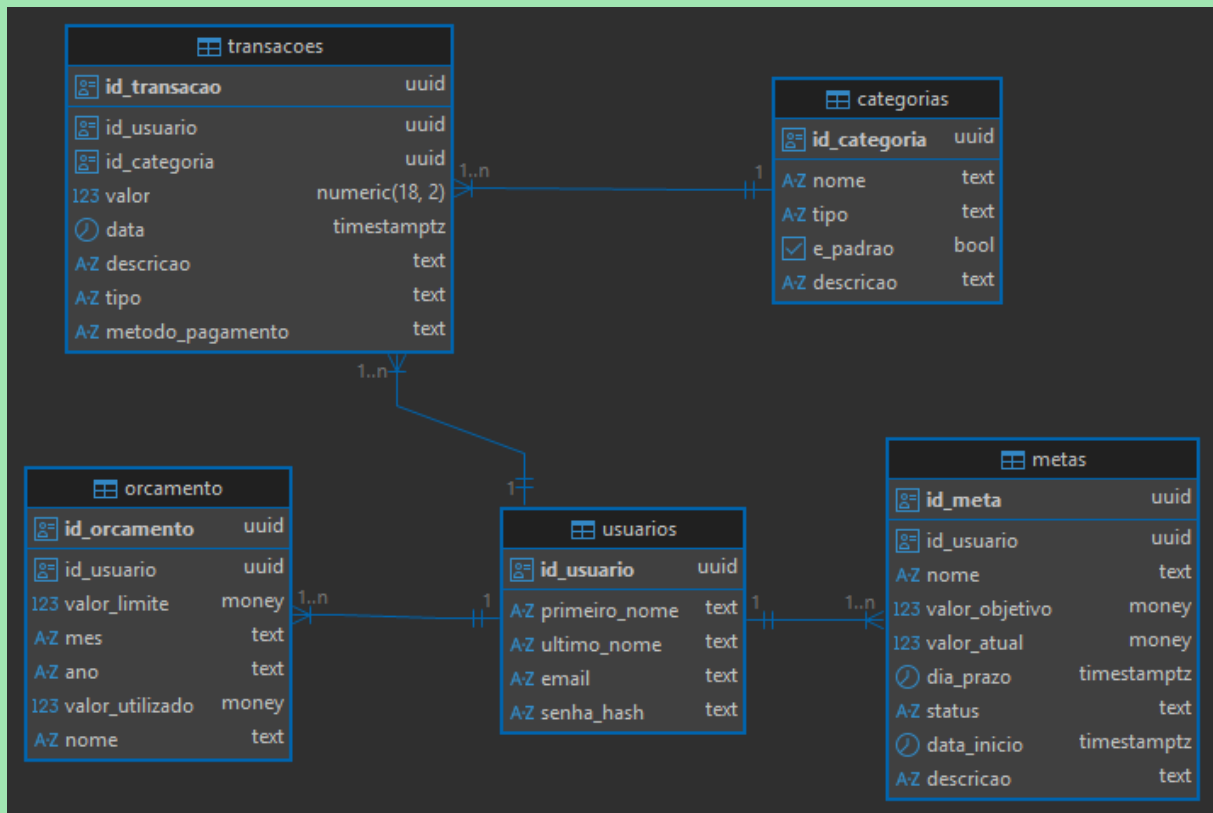
- A conversão deve usar a **taxa mais recente disponível** na API externa.

## DESIGN DO SISTEMA

### Diagrama de componentes



## Diagrama de Entidades (Banco de Dados)



## API ENDPOINTS

Classe	Metódo	HTTP request	Descrição
<i>CambiosApi</i>	<b>apiCambiosConverterMoedaGet</b>	<b>GET</b> /api/Cambios/convert er/{moeda}	Converte um valor em reais para a moeda especificada
<i>CambiosApi</i>	<b>apiCambiosCotacaoMoedaGet</b>	<b>GET</b> /api/Cambios/cotacao/{moeda}	Consulta a cotação atual de uma moeda (ARS, USD, EUR, CLP, UYU)
<i>CategoriasApi</i>	<b>apiCategoriasGet</b>	<b>GET</b> /api/Categorias	Retorna todas as categorias cadastradas.
<i>CategoriasApi</i>	<b>apiCategoriasIdDelete</b>	<b>DELETE</b> /api/Categorias/{id}	Remove uma categoria existente.
<i>CategoriasApi</i>	<b>apiCategoriasIdGet</b>	<b>GET</b> /api/Categorias/{id}	Retorna uma categoria específica pelo seu identificador único.
<i>CategoriasApi</i>	<b>apiCategoriasIdPut</b>	<b>PUT</b> /api/Categorias/{id}	Atualiza os dados de uma categoria existente.
<i>CategoriasApi</i>	<b>apiCategoriasPost</b>	<b>POST</b> /api/Categorias	Cria uma nova categoria.
<i>MetasApi</i>	<b>apiMetasGet</b>	<b>GET</b> /api/Metas	Retorna todas as metas cadastradas.
<i>MetasApi</i>	<b>apiMetasIdDelete</b>	<b>DELETE</b> /api/Metas/{id}	Remove uma meta existente.
<i>MetasApi</i>	<b>apiMetasIdGet</b>	<b>GET</b> /api/Metas/{id}	Retorna uma meta específica pelo seu identificador único.
<i>MetasApi</i>	<b>apiMetasIdPut</b>	<b>PUT</b> /api/Metas/{id}	Atualiza os dados de uma meta existente.
<i>MetasApi</i>	<b>apiMetasPost</b>	<b>POST</b> /api/Metas	Cria uma nova meta vinculada a um usuário.
<i>OrcamentosApi</i>	<b>apiOrcamentosGet</b>	<b>GET</b> /api/Orcamentos	Retorna todos os orçamentos cadastrados.
<i>OrcamentosApi</i>	<b>apiOrcamentosIdDelete</b>	<b>DELETE</b> /api/Orcamentos/{id}	Remove um orçamento pelo ID.





Classe	Metódo	HTTP request	Descrição
<i>OrcamentosApi</i>	<b>apiOrcamentosIdGet</b>	<b>GET</b> /api/Orcamentos/{id}	Retorna um orçamento específico pelo seu ID.
<i>OrcamentosApi</i>	<b>apiOrcamentosIdPut</b>	<b>PUT</b> /api/Orcamentos/{id}	Atualiza os dados de um orçamento existente.
<i>OrcamentosApi</i>	<b>apiOrcamentosPost</b>	<b>POST</b> /api/Orcamentos	Cria um novo orçamento.
<i>RelatoriosApi</i>	<b>apiRelatoriosUsuarioldDownloadGet</b>	<b>GET</b> /api/Relatorios/{usuariold}/download	Gera e baixa o relatório financeiro como arquivo JSON.
<i>RelatoriosApi</i>	<b>apiRelatoriosUsuarioldGet</b>	<b>GET</b> /api/Relatorios/{usuariold}	Gera um relatório financeiro para o usuário informado.
<i>TransacoesApi</i>	<b>apiTransacoesGet</b>	<b>GET</b> /api/Transacoes	Retorna todas as transações com paginação.
<i>TransacoesApi</i>	<b>apiTransacoesIdDelete</b>	<b>DELETE</b> /api/Transacoes/{id}	Remove uma transação pelo ID.
<i>TransacoesApi</i>	<b>apiTransacoesIdGet</b>	<b>GET</b> /api/Transacoes/{id}	Retorna uma transação pelo seu ID.
<i>TransacoesApi</i>	<b>apiTransacoesIdPut</b>	<b>PUT</b> /api/Transacoes/{id}	Atualiza uma transação existente.
<i>TransacoesApi</i>	<b>apiTransacoesPost</b>	<b>POST</b> /api/Transacoes	Cria uma nova transação.
<i>UsuariosApi</i>	<b>apiUsuariosGet</b>	<b>GET</b> /api/Usuarios	Retorna todos os usuários com paginação.
<i>UsuariosApi</i>	<b>apiUsuariosIdDelete</b>	<b>DELETE</b> /api/Usuarios/{id}	Remove um usuário pelo ID (hard delete).
<i>UsuariosApi</i>	<b>apiUsuariosIdGet</b>	<b>GET</b> /api/Usuarios/{id}	Retorna um usuário pelo ID.
<i>UsuariosApi</i>	<b>apiUsuariosIdPut</b>	<b>PUT</b> /api/Usuarios/{id}	Atualiza os dados de um usuário existente.
<i>UsuariosApi</i>	<b>apiUsuariosPost</b>	<b>POST</b> /api/Usuarios	Cria um novo usuário.

