Entrega 1 - Modelagem (Cheesman & Daniels)

Projeto FINTA - Topicos Avancados de Engenharia de Software
Fonte: entrega-1-secoes-1-4.md | Gerado em 25/02/2026

Documento consolidado com as secoes 1 a 4 da entrega original.

1) Escopo e premissas

- Escopo funcional validado:
  - Consultar preço do Bitcoin.
  - Consultar preço de carro na Tabela FIPE.
  - Adicionar ação à biblioteca (ex.: Google).
  - Cadastro de usuário (sem caso de uso de login nesta entrega).
- Ator primário: `Usuário`.
- Sistemas externos:
  - API de mercado financeiro.
  - API FIPE.
- Persistência da biblioteca do usuário no cliente (`localStorage`).
- Arquitetura alvo: `Frontend Web (Next/React) + Backend API (Elysia/Bun)`.

2) Diagrama de Casos de Uso

Trecho Mermaid

Bloco Mermaid omitido para priorizar legibilidade no PDF; consulte o arquivo fonte no Markdown.

Arquivo fonte: casos-de-uso.mmd

3) Descrição dos 3 Casos de Uso principais

UC-01 - Consultar preço do Bitcoin

- Objetivo: permitir que o usuário obtenha a cotação atual do Bitcoin.
- Atores: `Usuário` (primário), `API Mercado Financeiro` (secundário).
- Pré-condições: sistema disponível; integração com API de mercado ativa.
- Fluxo principal:
  1. Usuário solicita consulta de preço do Bitcoin.
  2. Frontend envia requisição ao Backend.
  3. Backend consulta a API de mercado financeiro.
  4. Backend normaliza o payload e retorna ao Frontend.
  5. Frontend exibe preço, data/hora da cotação e fonte.
- Fluxos alternativos:
  - API externa indisponível: Backend retorna erro técnico padronizado.
  - Timeout: Backend retorna fallback de indisponibilidade temporária.
- Pós-condições: cotação exibida ao usuário ou mensagem de falha registrada.

UC-02 - Consultar preço de carro na Tabela FIPE

- Objetivo: permitir consulta de preço de referência FIPE por veículo.
- Atores: `Usuário` (primário), `API FIPE` (secundário).
- Pré-condições: dados mínimos do veículo informados (`marca`, `modelo`, `ano`).
- Fluxo principal:
  1. Usuário informa marca, modelo e ano.
  2. Frontend chama Backend com os parâmetros.
  3. Backend consulta a API FIPE.

4. Backend valida e mapeia resultado para contrato interno.
5. Frontend apresenta valor FIPE e metadados da consulta.

- Fluxos alternativos:
  - Veículo não encontrado: retorno de resultado vazio com mensagem orientativa.
  - API FIPE indisponível: retorno de erro técnico padronizado.
- Pós-condições: valor FIPE exibido ou falha comunicada de forma consistente.

UC-03 - Adicionar ação à biblioteca (ex.: Google)

- Objetivo: permitir salvar um ativo de interesse na biblioteca pessoal (exemplo: ação do Google).
- Atores: `Usuário` (primário), `API Mercado Financeiro` (secundário).
- Pré-condições: usuário cadastrado (cadastro simples); navegador com armazenamento habilitado.
- Fluxo principal:
  1. Usuário seleciona uma ação para adicionar (exemplo: Google).
  2. Frontend requisita cotação atual via Backend.
  3. Backend consulta API de mercado e retorna dados consolidados.
  4. Frontend persiste ativo na biblioteca local do usuário.
  5. Frontend confirma inclusão e atualiza a lista.
- Fluxos alternativos:
  - Ativo já existente na biblioteca: sistema evita duplicidade e informa usuário.
  - Falha de persistência local: sistema informa erro e não confirma inclusão.
- Pós-condições: biblioteca local contém a ação (sem duplicidade) ou operação falha de forma explícita.

4) Interfaces de software

4.1 Interfaces externas fornecidas pelo FINTA

- `IPriceQueryService`
  - `getCryptoQuote(symbol: string): CryptoQuote`
  - `getStockQuote(ticker: string): StockQuote`
- `IFipeQueryService`
  - `getVehiclePrice(input: VehicleQuery): FipePrice`
- `IRegistrationService`
  - `register(input: RegistrationInput): RegistrationResult`
- `IUserLibraryService`
  - `addAsset(input: AssetInput): LibraryItem`
  - `listAssets(): LibraryItem[]`

4.2 Interfaces externas requeridas pelo FINTA

- `IMarketDataGateway` (API de mercado financeiro)
  - `fetchCrypto(symbol: string): ExternalCryptoQuote`
  - `fetchStock(ticker: string): ExternalStockQuote`
- `IFipeGateway` (API FIPE)
  - `fetchVehiclePrice(input: VehicleQuery): ExternalFipePrice`

4.3 Interfaces internas entre componentes

- `IAuthRepository`

    - createUser(input: RegistrationInput): User
    - existsByEmail(email: string): boolean
- IUserLibraryRepository
    - save(item: LibraryItem): LibraryItem
    - findAll(): LibraryItem[]
    - existsByTicker(ticker: string): boolean