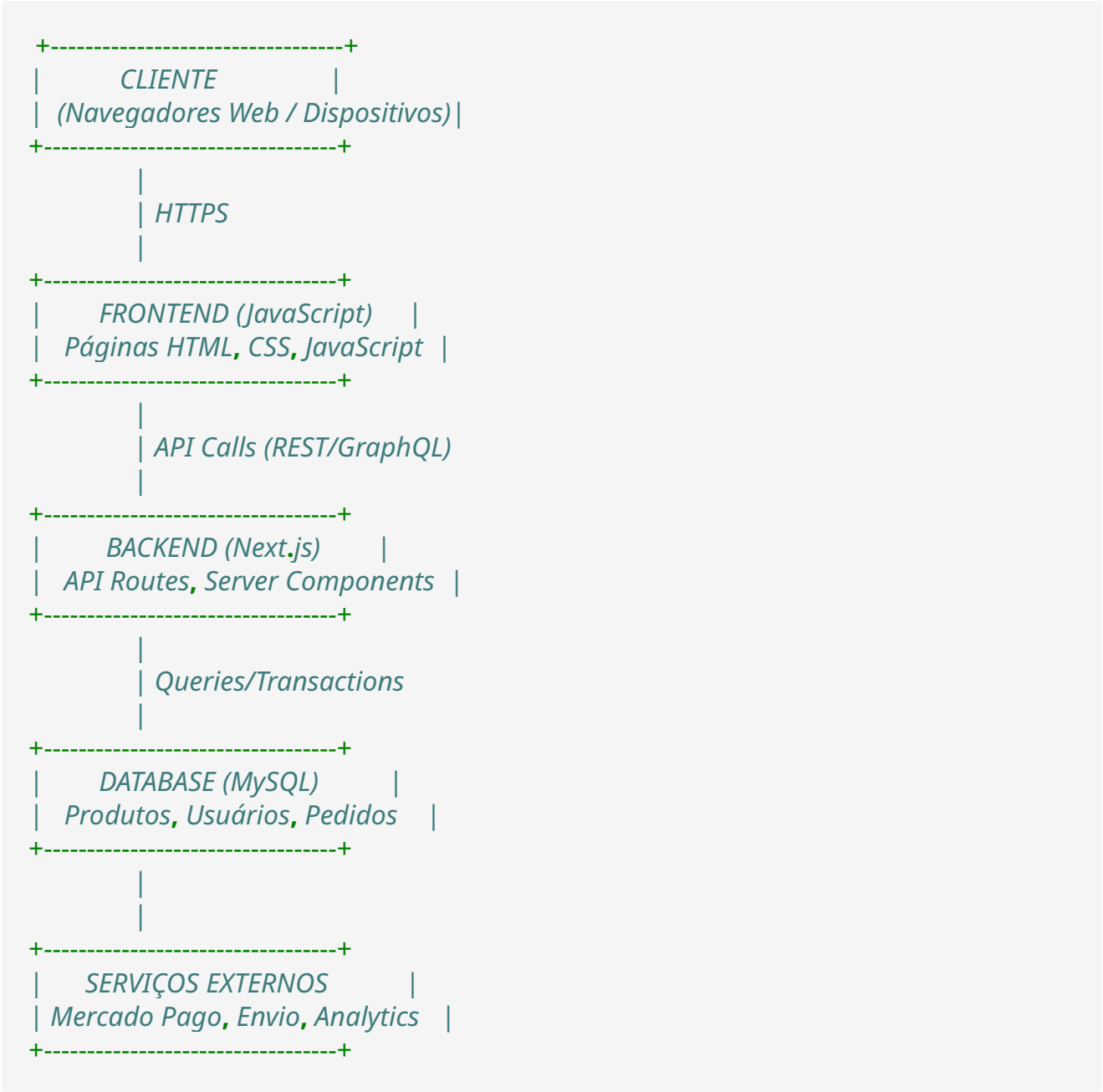
Arquitetura e Tecnologias para o E-commerce

Visão Geral da Arquitetura

A arquitetura do e-commerce será baseada em um modelo moderno de aplicação web, seguindo os princípios de desenvolvimento escalável, seguro e orientado a componentes. Adotaremos uma abordagem que separa claramente o frontend do backend, permitindo desenvolvimento independente e manutenção simplificada.

Diagrama de Arquitetura



Tecnologias Selecionadas

Frontend

1. **JavaScript Puro**
2. Conforme solicitado pelo cliente, utilizaremos JavaScript vanilla para o desenvolvimento do frontend
3. Estrutura baseada em componentes reutilizáveis
4. Uso de módulos ES6 para organização do código
5. Implementação de padrões de design como Observer, Factory e Singleton
6. **HTML5 & CSS3**
7. HTML5 semântico para melhor acessibilidade e SEO
8. CSS3 com variáveis para consistência visual
9. Flexbox e Grid para layouts responsivos
10. Media queries para adaptação a diferentes dispositivos
11. **Bibliotecas Auxiliares**
12. SwiperJS para carrosséis e sliders
13. LightboxJS para visualização de imagens
14. ValidatorJS para validação de formulários
15. MomentJS para manipulação de datas
16. **Ferramentas de Build**
17. Webpack para bundling e otimização
18. Babel para compatibilidade com navegadores antigos
19. PostCSS para processamento de CSS
20. ESLint e Prettier para qualidade de código

Backend

1. **Next.js**
2. Framework React com renderização do lado do servidor (SSR)
3. API Routes para endpoints de backend
4. Suporte a Server Components
5. Otimização automática de imagens e fontes
6. Roteamento baseado em sistema de arquivos

7. **Node.js**

8. Ambiente de execução JavaScript no servidor
9. Gerenciamento de pacotes via npm/yarn

10. Suporte a ES6+ e TypeScript

11. **Bibliotecas de Backend**

12. Prisma como ORM para interação com o banco de dados
13. NextAuth.js para autenticação e autorização
14. Multer para upload de arquivos
15. Zod para validação de dados
16. Winston para logging

Banco de Dados

1. **MySQL**
2. Sistema de gerenciamento de banco de dados relacional
3. Suporte a transações ACID
4. Índices para otimização de consultas
5. Procedimentos armazenados para lógica complexa

6. Triggers para manutenção da integridade dos dados

7. **Estratégia de Dados**

8. Normalização até a 3ª forma normal
9. Uso de chaves estrangeiras para relacionamentos
10. Índices compostos para consultas frequentes
11. Particionamento para tabelas grandes

Integração de Pagamentos

1. **Mercado Pago**
2. SDK oficial do Mercado Pago para JavaScript
3. Implementação de Checkout Pro para experiência completa
4. Checkout Transparente para experiência personalizada
5. Webhooks para notificações de pagamento
6. Gestão de reembolsos e cancelamentos

Infraestrutura e DevOps

1. **Hospedagem e Deployment**

2. Vercel para hospedagem do Next.js
3. AWS RDS para o banco de dados MySQL
4. CloudFlare para CDN e proteção DDoS

5. CI/CD via GitHub Actions

6. **Monitoramento e Análise**

7. Google Analytics para análise de tráfego
8. Hotjar para mapas de calor e gravações de sessão
9. Sentry para monitoramento de erros
10. Uptime Robot para monitoramento de disponibilidade

Padrões de Arquitetura e Design

Padrões de Arquitetura

1. **Arquitetura em Camadas**
2. Camada de Apresentação (Frontend)
3. Camada de Aplicação (Lógica de Negócios)
4. Camada de Dados (Acesso ao Banco de Dados)

5. Camada de Infraestrutura (Serviços Externos)

6. **Padrão MVC (Model-View-Controller)**

7. Model: Representação dos dados e regras de negócio
8. View: Interface do usuário

9. Controller: Gerencia a entrada do usuário e coordena Model e View

10. **API RESTful**

11. Endpoints organizados por recursos
12. Uso apropriado dos métodos HTTP (GET, POST, PUT, DELETE)
13. Respostas com códigos de status HTTP adequados
14. Documentação via Swagger/OpenAPI

Padrões de Design

1. **Componentes Reutilizáveis**
2. Botões, formulários, cards, modais
3. Componentes de navegação e layout

4. Componentes de listagem e detalhamento de produtos

5. **Atomic Design**

6. Átomos: elementos básicos (botões, inputs)
7. Moléculas: grupos de átomos (formulários de busca)
8. Organismos: grupos de moléculas (cabeçalho, rodapé)
9. Templates: estruturas de página

10. Páginas: instâncias específicas de templates

11. **Padrões de Estado**

12. Gerenciamento de estado global para carrinho e usuário
13. Estados locais para componentes específicos
14. Persistência de estado via localStorage/sessionStorage

Considerações de Segurança

1. **Autenticação e Autorização**
2. Autenticação baseada em JWT (JSON Web Tokens)
3. Autorização baseada em funções (RBAC)
4. Proteção contra ataques de força bruta

5. Implementação de 2FA (autenticação de dois fatores)

6. **Segurança de Dados**

7. Criptografia de dados sensíveis
8. Sanitização de inputs para prevenir injeção SQL
9. Proteção contra XSS (Cross-Site Scripting)

10. Implementação de CSP (Content Security Policy)

11. **Conformidade e Privacidade**

12. Conformidade com LGPD (Lei Geral de Proteção de Dados)
13. Política de privacidade clara e acessível
14. Consentimento explícito para coleta de dados
15. Opção de exclusão de dados do usuário

Escalabilidade e Performance

1. **Estratégias de Escalabilidade**

2. Arquitetura stateless para facilitar escalabilidade horizontal
3. Caching em múltiplos níveis (browser, CDN, servidor)
4. Otimização de consultas ao banco de dados
5. Lazy loading de componentes e imagens
6. **Otimização de Performance**
7. Minificação e compressão de assets
8. Code splitting para carregamento sob demanda
9. Otimização de imagens (WebP, dimensionamento automático)
10. Implementação de Service Workers para experiência offline

Estrutura de Diretórios

Frontend (JavaScript Puro)

```
/frontend
 /assets
   /images
   /fonts
   /icons
 /css
   /components
   /pages
   /utils
   main.css
 /js
   /components
     /common
     /product
     /cart
     /checkout
     /user
   /pages
     home.js
     product-list.js
     product-detail.js
     cart.js
     checkout.js
     account.js
   /services
     api.js
     auth.js
     cart.js
     payment.js
   /utils
```

```
    formatter.js
    validator.js
    storage.js
  main.js
 /templates
  header.html
  footer.html
  product-card.html
 index.html
```

Backend (Next.js)

```
/backend
 /src
  /app
   /api
    /auth
     [...nextauth]/route.js
    /products
     route.js
    /cart
     route.js
    /orders
     route.js
    /webhooks
     mercadopago/route.js
   /admin
    /dashboard
     page.js
    /products
     page.js
    /orders
     page.js
   /product
    /[id]
     page.js
   /category
    /[slug]
     page.js
   /cart
    page.js
   /checkout
    page.js
   /account
    page.js
   page.js
   layout.js
  /components
   /ui
   /layout
```

```
    /product
    /cart
    /checkout
  /lib
    /prisma
    /auth
    /mercadopago
  /models
  /utils
/prisma
  schema.prisma
/public
next.config.js
package.json
```

Conclusão

A arquitetura proposta para o e-commerce foi projetada para atender aos requisitos específicos do cliente, utilizando JavaScript puro no frontend, Next.js no backend e MySQL como banco de dados. Esta estrutura oferece um equilíbrio entre simplicidade, performance e escalabilidade, permitindo a implementação das tendências de mercado identificadas na pesquisa anterior.

A separação clara entre frontend e backend, juntamente com a adoção de padrões de design estabelecidos, facilitará o desenvolvimento, manutenção e evolução do sistema ao longo do tempo. A integração com o Mercado Pago garantirá uma experiência de pagamento segura e confiável para os usuários.

As considerações de segurança, escalabilidade e performance foram incorporadas desde o início do projeto, garantindo que o e-commerce possa crescer e se adaptar às necessidades futuras do negócio.