



# Shopify App Project Structure

Remix-based Shopify app with Polaris UI, App Bridge, and GraphQL Admin API integration.

Updated 2025-12-22

#shopify #app #remix #ecommerce #typescript

PNG

PDF

Copy

Prompt

## Project Directory

### my-shopify-app/

- app/ Remix app direc...
  - routes/ Route modules
    - \_index/ App home page
      - app.\_index.tsx Main app route
      - app.settings.tsx Settings page
      - auth.\$.tsx OAuth callback ...
      - webhooks.tsx Webhook handlers
    - components/ Reusable UI com...
      - ProductList.tsx
      - SettingsForm.tsx
    - graphql/ GraphQL queries...
      - queries.ts
      - mutations.ts
    - utils/
      - shopify.server.ts Shopify client
      - root.tsx Root layout wit...
      - shopify.server.ts Shopify API set...
  - prisma/ Database schema
    - schema.prisma Session storage
  - extensions/ App extensions
    - theme-extension/ Theme app exten...
      - blocks/
        - app-block.liquid
        - shopify.extension.toml
    - shopify.app.toml App configurati...
    - package.json
    - remix.config.js
    - .env API keys (gitig...
    - .gitignore
    - README.md

## Why This Structure?

This structure uses Shopify's official Remix template. Routes handle OAuth, webhooks, and app pages. Polaris provides the UI. Prisma stores sessions. The `extensions/` folder contains theme extensions that run on the storefront.

## Key Directories

**app/routes/** - Remix routes for app pages, auth, webhooks

**app/graphql/** - Admin API queries and mutations

**prisma/** - Database schema for session storage

**extensions/** - Theme and checkout extensions

## Basic App Route

```
// app/routes/app._index.tsx
import { Page, Card, Text } from "@shopify/polaris";
import { useLoaderData } from "@remix-run/react";
import { authenticate } from "../shopify.server";

export async function loader({ request }) {
  const { admin } = await authenticate.admin(request);
  const response = await admin.graphql(`{ shop { name } }`);
  return json({ shop: response.data.shop });
}

export default function Index() {
  const { shop } = useLoaderData();
  return <Page title={shop.name}><Card>...</Card></Page>;
}
```

## Getting Started

- `npm init @shopify/app@latest`
- Select Remix template
- `cd my-app && npm install`
- `npm run dev` to start with Shopify CLI
- Install app on development store

## Shopify APIs

**Admin API** - GraphQL API for store data (products, orders)

**App Bridge** - Embed app in Shopify admin UI

**Polaris** - React components matching Shopify's design

**Webhooks** - React to store events (orders, products)

## Best Practices

- Use `authenticate.admin()` for all authenticated routes
- Store sessions in database, not memory
- Handle rate limits with retry logic
- Use App Bridge for navigation within Shopify admin
- Register webhooks in `shopify.app.toml`

## Extension Types

**Theme Extension** - App blocks for Online Store 2.0 themes

**Checkout Extension** - Customize checkout UI

**Function** - Server-side logic for discounts, shipping

## When To Use This

- Building merchant-facing admin tools
- Extending storefront with app blocks
- Custom checkout experiences
- Automating store operations
- Integrating third-party services

## Trade-offs

**Shopify CLI required** - Development requires Shopify CLI tooling

**Partner account** - Need Shopify Partner account for development

**Review process** - Public apps require Shopify approval