

Project Details

Name: **ADA Payments in the Credit Card Machine integrated in Merchants & withdraw \$ADA <> Fiat BRL on bank ATMs**

Catalyst ID: # 1200013 - Fund: Fund 12

Proposer name: Otávio (Cardano Feed) - Email: contato@otaviolima.com

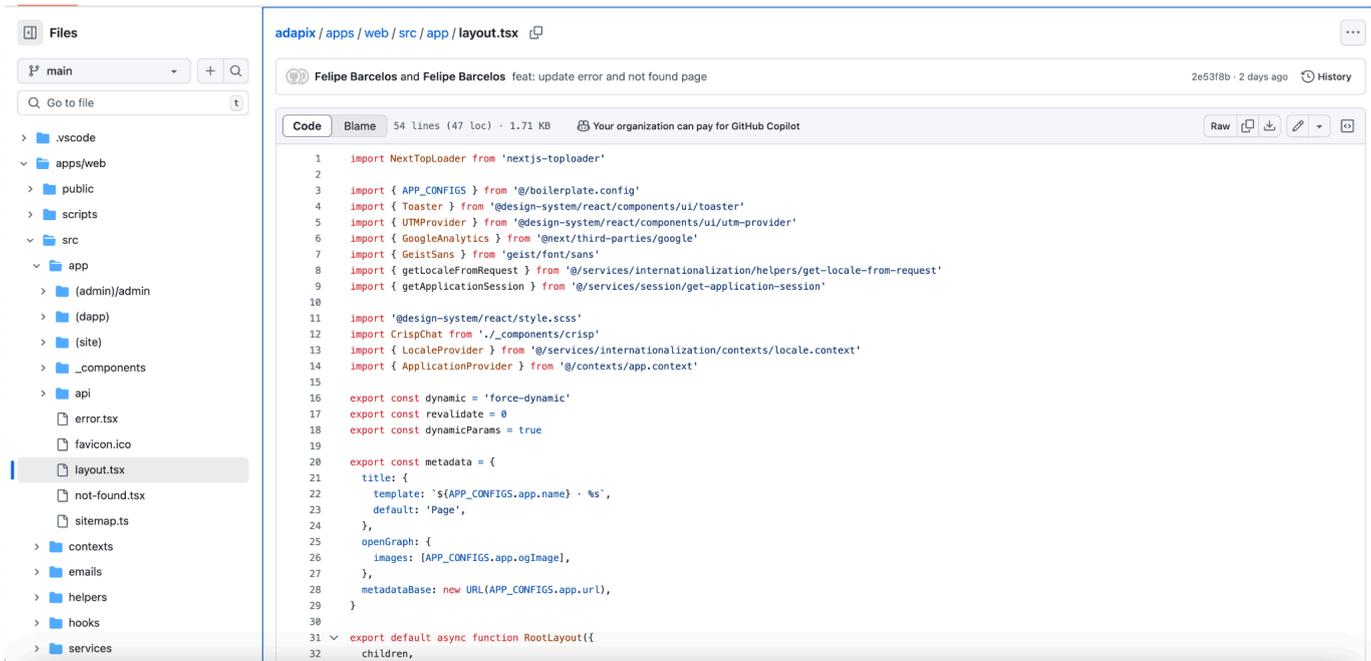
Front-end and Back-end programming

Document Version: 1.0 - Date: July 20, 2025.

Introduction

This document maps out the **Front-end and Back-end** programming for **ADA Payments**, a web-based platform that serves as a gateway to trading Cardano (ADA) cryptocurrency.

1. Front-end screenshots



Files

main

Go to file

- public
- scripts
- src
 - app
 - (admin)/admin
 - (dapp)
 - (site)
 - __components
 - __data
 - faq.ts
 - README.md
 - actions.tsx
 - layout.tsx
 - page.tsx
 - __components
 - crisp.tsx
 - logo.tsx
 - api
 - error.tsx
 - favicon.ico
 - layout.tsx

adapix / apps / web / src / app / (site) / actions.tsx

Felipe Barcelos and Felipe Barcelos first commit a8bd9c1 · 2 days ago History

Code Blame 17 lines (15 loc) · 407 Bytes Your organization can pay for GitHub Copilot

```
1 import { client } from '@services/actions/public-client'
2 import { modules } from '@app/modules/src'
3 import { cache } from 'react'
4
5 export const getAdaQuoteAction = cache(
6   client.action({
7     name: 'ada.get-quote',
8     type: 'query',
9     handler: async () => {
10       const adaPrice = await modules.provider.market.getQuote('cardano', 'brl')
11     }
12   })
13 )
14
15 },
16 },
17 )
```

Files

main

Go to file

- public
- scripts
- src
 - app
 - (admin)/admin
 - (dapp)
 - (site)
 - __components
 - __data
 - faq.ts
 - README.md
 - actions.tsx
 - layout.tsx
 - page.tsx
 - __components
 - crisp.tsx
 - logo.tsx
 - api
 - error.tsx
 - favicon.ico
 - layout.tsx

adapix / apps / web / src / app / (site) / layout.tsx

Felipe Barcelos and Felipe Barcelos feat: update errors on revalidate f8b02a6 · 1 day ago History

Code Blame 13 lines (12 loc) · 303 Bytes Your organization can pay for GitHub Copilot

```
1 import { PropsWithChildren } from 'react'
2 import { Footer } from '../_components/footer'
3 import { Header } from '../_components/header'
4
5 export default function Layout({ children }: PropsWithChildren) {
6   return (
7     <div>
8       <Header />
9       <main>{children}</main>
10      <Footer />
11    </div>
12  )
13 }
```

The screenshot shows a code editor interface. On the left, a file explorer shows a directory structure: `src` (main), `app` (main), `(admin)/admin`, `(dapp)`, `(site)`, `_components`, `_data` (containing `README.md`, `actions.tsx`, `layout.tsx`, `page.tsx`), `_components` (containing `crisp.tsx`, `logo.tsx`), `api` (containing `error.tsx`, `favicon.ico`, `layout.tsx`), `not-found.tsx` (selected), `sitemap.ts`, and `contexts`. The main editor area shows the code for `not-found.tsx`. The commit message is "Felipe Barcelos and Felipe Barcelos feat: update error and not found page" from 2 days ago. The code includes imports for `Link`, `Logo`, `Button`, `ArrowLeft`, and `Metadata`. It defines a `metadata` object with title and description. The `NotFound` function returns a JSX element with a logo, a heading, a paragraph, and a button.

```
1 import Link from 'next/link'
2 import { Logo } from '../components/logo'
3 import { Button } from '@design-system/react/components/ui/button'
4 import { ArrowLeft } from 'lucide-react'
5 import type { Metadata } from 'next'
6
7 /**
8  * Metadata for the Not Found page
9  */
10 export const metadata: Metadata = {
11   title: 'Página não encontrada',
12   description:
13     'Desculpe, não conseguimos encontrar a página que você está procurando.',
14 }
15
16 /**
17  * Custom 404 Not Found page component
18  */
19 export default function NotFound() {
20   return (
21     <div className="min-h-screen flex flex-col items-center justify-center gap-8">
22       <Logo className="h-12" />
23
24       <div className="max-w-md text-center space-y-2">
25         <h1 className="text-4xl font-bold">Página não encontrada</h1>
26         <p className="text-muted-foreground">
27           Desculpe, não conseguimos encontrar a página que você está procurando.
28         </p>
29       </div>
30
31       <Button asChild variant="outline">
32         <Link href="/" className="gap-2">
```

The screenshot shows a code editor interface. On the left, a file explorer shows a directory structure: `public`, `scripts`, `src` (main), `app` (main), `(admin)/admin`, `(dapp)`, `(site)`, `_components` (containing `dapp.tsx` (selected), `faq-section-ada.tsx`, `footer.tsx`, `header.tsx`, `hero-section.tsx`, `how-to-works-section.tsx`, `page-wrapper.tsx`), `_data` (containing `README.md`, `actions.tsx`, `layout.tsx`, `page.tsx`), and `_components` (containing `crien.tsx`). The main editor area shows the code for `dapp.tsx`. The commit message is "Felipe Barcelos and Felipe Barcelos first commit" from 1 day ago. The code imports `getUrl` and `getApplicationSession`. The `Dapp` function is an async function that gets the session and pathname, checks for KYC status, and returns an `iframe` element.

```
1 import { getUrl } from '@helpers/get-url'
2 import { getApplicationSession } from '@services/session/get-application-session'
3
4 export async function Dapp() {
5   // Get the session from the server
6   const session = await getApplicationSession()
7
8   // Get the pathname from the headers
9   let dappPath = '/dapp'
10
11   if (session.user) {
12     // Check if the user has KYC approved
13     const isKycPending = session.user.settings.kyc.status === 'pending'
14     const isKycSubmitted = session.user.settings.kyc.status === 'submitted'
15
16     // Redirect to home page if user is on the KYC page and KYC is approved
17     if (isKycSubmitted) dappPath = '/dapp/kyc/validate-user-data'
18     if (isKycPending) dappPath = '/dapp/kyc'
19   }
20
21   return (
22     <div className="bg-card border border-border rounded-3xl h-[44rem] sticky top-20 overflow-hidden">
23       <iframe
24         src=getUrl(dappPath)
25         className="w-full h-full relative overflow-hidden rounded-3xl"
26         allow="clipboard-write; usb; camera; microphone; cookies; headers; *; web3"
27         title=""
28       ></iframe>
29     </div>
30   )
31 }
```

2. Back-end screenshots

The screenshot shows a GitHub code viewer interface. On the left, a file explorer shows the directory structure: (admin)/admin, (main), _components, _types, _utils, transactions, users, README.md, layout.tsx, (dapp), (site), _components, api, error.tsx, favicon.ico, layout.tsx, not-found.tsx, sitemap.ts, contexts, emails, and helpers. The main area displays the code for layout.tsx, which is 30 lines long (26 loc) and 721 bytes. The code includes imports for AdminSidebar, redirect, and getCurrentUser, and defines an AdminLayout component that checks user roles and renders a sidebar and main content area.

```
1 import { AdminSidebar } from '../_components/layout/sidebar'
2 import { redirect } from 'next/navigation'
3 import { getCurrentUser } from '@services/session/get-current-user'
4
5 export const dynamic = 'force-dynamic'
6 export const fetchCache = 'force-no-store'
7 export const revalidate = 0
8 export const dynamicParams = true
9
10 /**
11  * Admin layout component
12  */
13 export default async function AdminLayout({
14   children,
15 }): {
16   children: React.ReactNode
17 } {
18   const user = await getCurrentUser()
19
20   if (!user || user.role !== 'ADMIN') {
21     redirect('/')
22   }
23
24   return (
25     <div className="min-h-screen flex bg-background">
26       <AdminSidebar users={user} />
27       <main className="flex-1 h-full">{children}</main>
28     </div>
29   )
30 }
```

The screenshot shows a GitHub code viewer interface. On the left, a file explorer shows the directory structure: .vscode, apps/web, public, scripts, src, app, (admin)/admin, (main), _components, _types, _utils, transactions, _components, README.md, actions.ts, page.tsx, schemas.ts, users, README.md, layout.tsx, and (dapp). The main area displays the code for page.tsx, which is 57 lines long (51 loc) and 1.54 KB. The code imports various components like DashboardPage, DataTable, and TransactionFilters, and defines a TransactionsPage component that handles search parameters and displays a list of transactions.

```
1 import {
2   DashboardPage,
3   DashboardPageHeader,
4   DashboardPageMain,
5 } from '@design-system/react/components/shared/dashboard/page'
6 import { DataTable } from '../_components/data-display/data-table'
7 import { getTransactions } from './actions'
8 import { columns } from '../_components/columns'
9 import { TransactionFilters } from '../_components/transaction-filters'
10 import type { Metadata } from 'next'
11
12 export const dynamic = 'force-dynamic'
13 export const fetchCache = 'force-no-store'
14 export const revalidate = 0
15 export const dynamicParams = true
16
17 /**
18  * Props for the TransactionsPage component
19  */
20 interface TransactionsPageProps {
21   searchParams: {
22     search?: string
23     status?: string
24     type?: string
25   }
26 }
27
28 /**
29  * Transactions page component that displays a list of transactions in a data table
30  */
31 export default async function TransactionsPage({
32   searchParams,
```

Files

- main
- public
- scripts
- src
 - app
 - (admin)/admin
 - (main)
 - _components
 - _types
 - _utils
 - transactions
 - _components
 - actions.ts
 - page.tsx
 - schemas.ts
 - users
 - README.md
 - layout.tsx
 - (dapp)
 - (site)
 - _components

adapix / apps / web / src / app / (admin) / admin / transactions / actions.ts

Felipe Barcelos and Felipe Barcelos feat: update admin and kyc improvements 536c271 · 2 days ago History

Code Blame 141 Lines (133 loc) · 3.07 KB Your organization can pay for GitHub Copilot

```
1 'use server'
2
3 import { db } from '@app/db'
4 import { z } from 'zod'
5 import type { Transaction } from '../_components/columns'
6 import { client } from '@services/actions/admin-client'
7
8 /**
9  * Get all transactions from the database with filters
10  */
11 export const getTransactions = client.action({
12   name: 'getTransactions',
13   type: 'query',
14   schema: z.object({
15     search: z.string().optional(),
16     status: z.string().optional(),
17     type: z.string().optional(),
18   }),
19   handler: async ({ input }) => {
20     const transactions = await db.transaction.findMany({
21       where: {
22         AND: [
23           // Search filter
24           input.search
25           ? {
26             OR: [
27               {
28                 user: {
29                   name: { contains: input.search, mode: 'insensitive' },
30                 },
31               },
32             ],
```

Files

- main
- public
- scripts
- src
 - app
 - (admin)/admin
 - (main)
 - _components
 - _types
 - _utils
 - transactions
 - _components
 - columns.tsx
 - transaction-filters.tsx
 - transaction-sheet.tsx
 - transaction-table.tsx
 - users
 - README.md
 - (dapp)
 - (site)
 - _components

adapix / apps / web / src / app / (admin) / admin / transactions / _components / transaction-table.tsx

Felipe Barcelos and Felipe Barcelos feat: add admin 34375d7 · 2 days ago History

Code Blame 18 Lines (15 loc) · 460 Bytes Your organization can pay for GitHub Copilot

```
1 'use client'
2
3 import { DataTable } from '../../_components/data-display/data-table'
4 import { columns, Transaction } from '../columns'
5
6 /**
7  * TransactionTable component props
8  */
9 interface TransactionTableProps {
10   transactions: Transaction[]
11 }
12
13 /**
14  * TransactionTable component that displays transactions in a table
15  */
16 export function TransactionTable({ transactions }: TransactionTableProps) {
17   return <DataTable columns={columns} data={transactions} />
18 }
```

The screenshot shows a code editor interface. On the left is a file explorer with a search bar and a list of files and folders. The main area is a code editor showing the content of 'page.tsx'. The code includes imports for 'DashboardPage', 'DashboardPageHeader', and 'DashboardPageMain'. It also imports 'DataTable' from a shared component, 'getTransactions' from './actions', 'columns' from './components/columns', and 'TransactionFilters' from './components/transaction-filters'. There are several constant exports for 'dynamic', 'fetchCache', 'revalidate', and 'dynamicParams'. An interface 'TransactionsPageProps' is defined with properties 'searchParams', 'search?', 'status?', and 'type?'. Finally, an 'export default async function TransactionsPage' is shown, starting with 'searchParams'.

```
1 import {
2   DashboardPage,
3   DashboardPageHeader,
4   DashboardPageMain,
5 } from '@design-system/react/components/shared/dashboard/page'
6 import { DataTable } from '../_components/data-display/data-table'
7 import { getTransactions } from './actions'
8 import { columns } from './_components/columns'
9 import { TransactionFilters } from './_components/transaction-filters'
10 import type { Metadata } from 'next'
11
12 export const dynamic = 'force-dynamic'
13 export const fetchCache = 'force-no-store'
14 export const revalidate = 0
15 export const dynamicParams = true
16
17 /**
18  * Props for the TransactionsPage component
19  */
20 interface TransactionsPageProps {
21   searchParams: {
22     search?: string
23     status?: string
24     type?: string
25   }
26 }
27
28 /**
29  * Transactions page component that displays a list of transactions in a data table
30  */
31 export default async function TransactionsPage({
32   searchParams,
```