Transcrição do Projeto

e2e\demo.test.ts

```
import { expect, test } from '@playwright/test'; test('home page has expected h1', async ({ page }) =>
{ await page.goto('/'); await expect(page.locator('h1')).toBeVisible(); });
```

eslint.config.js

```
import prettier from 'eslint-config-prettier'; import { includeIgnoreFile } from '@eslint/compat';
import js from '@eslint/js'; import svelte from 'eslint-plugin-svelte'; import globals from 'globals';
import { fileURLToPath } from 'node:url'; import ts from 'typescript-eslint'; import svelteConfig from
'./svelte.config.js'; const gitignorePath = fileURLToPath(new URL('./.gitignore', import.meta.url));
export default ts.config( includeIgnoreFile(gitignorePath), js.configs.recommended,
...ts.configs.recommended, ...svelte.configs.recommended, prettier, ...svelte.configs.prettier, {
languageOptions: { globals: { ...globals.browser, ...globals.node } }, rules: { // typescript-eslint
strongly recommend that you do not use the no-undef lint rule on TypeScript projects. // see:
https://typescript-eslint.io/troubleshooting/faqs/eslint/#i-get-errors-from-the-no-undef-rule-about-global-variab
'no-undef': 'off' } }, { files: ['**/*.svelte', '**/*.svelte.ts', '**/*.svelte.js'], languageOptions:
{ parserOptions: { projectService: true, extraFileExtensions: ['.svelte'], parser: ts.parser,
svelteConfig } } } );
```

hooks.server.ts

```
// File: src/hooks.server.ts (COMPATÍVEL COM LUCIA v2.x) import { lucia } from '$lib/server/auth';
import type { Handle } from '@sveltejs/kit'; export const handle: Handle = async ({ event, resolve })
=> { // Tenta obter o ID da sessão a partir do cookie const sessionId =
event.cookies.get(lucia.sessionCookieName); // Se não houver session ID, o usuário não está logado if
(!sessionId) { event.locals.user = null; event.locals.session = null; return resolve(event); } //
Valida a sessão usando o Lucia const { session, user } = await lucia.validateSession(sessionId); // Se
a sessão for nova (fresh), atualiza o cookie no navegador if (session && session.fresh) { const
sessionCookie = lucia.createSessionCookie(session.id); event.cookies.set(sessionCookie.name,
sessionCookie.value, { path: '.', ...sessionCookie.attributes }); } // Se a sessão for inválida,
remove o cookie do navegador if (!session) { const sessionCookie = lucia.createBlankSessionCookie();
event.cookies.set(sessionCookie.name, sessionCookie.value, { path: '.', ...sessionCookie.attributes
}); } // Disponibiliza o `user` e `session` para `event.locals` event.locals.user = user;
event.locals.session = session; // Continua o fluxo da requisição return resolve(event); };
```

package.json

```
{ "name": "diario-de-trade-v2", "private": true, "version": "0.0.1", "type": "module", "scripts": {
"dev": "vite dev", "build": "vite build", "preview": "vite preview", "prepare": "svelte-kit sync ||
echo ''", "check": "svelte-kit sync && svelte-check --tsconfig ./tsconfig.json", "check:watch":
"svelte-kit sync && svelte-check --tsconfig ./tsconfig.json --watch", "format": "prettier --write .",
"lint": "prettier --check . && eslint .", "test:unit": "vitest", "test": "npm run test:unit -- --run
&& npm run test:e2e", "test:e2e": "playwright test" }, "dependencies": { "@lucia-auth/adapter-prisma":
"^4.0.1", "@prisma/client": "^5.22.0", "clsx": "^2.1.1", "lucia": "^3.2.2", "oslo": "^1.2.1",
"svelte-i18n": "^4.0.1", "sveltekit-superforms": "2.0.0-alpha.53", "tailwind-merge": "^2.6.0",
"tailwind-variants": "^0.1.20", "zod": "^3.25.76" }, "devDependencies": { "@eslint/compat": "^1.2.5",
"@eslint/js": "^9.18.0", "@playwright/test": "^1.49.1", "@sveltejs/adapter-auto": "^6.0.0",
"@sveltejs/kit": "^2.25.1", "@sveltejs/vite-plugin-svelte": "^6.0.0", "@tailwindcss/forms": "^0.5.10",
"@tailwindcss/postcss": "^4.1.11", "@tauri-apps/cli": "^2.6.2", "@types/node": "^20.19.8",
"autoprefixer": "^10.4.21", "daisyui": "^4.12.24", "eslint": "^9.18.0", "eslint-config-prettier":
"^10.0.1", "eslint-plugin-svelte": "^3.0.0", "globals": "^16.0.0", "playwright": "^1.53.0", "postcss":
"^8.5.6", "prettier": "^3.4.2", "prettier-plugin-svelte": "^3.3.3", "prisma": "^5.22.0", "svelte":
"^5.36.7", "svelte-check": "^4.0.0", "typescript": "^5.4.5", "typescript-eslint": "^8.20.0", "vite":
"^7.0.4", "vitest": "^3.2.3", "vitest-browser-svelte": "^0.1.0" } }
```

playwright.config.ts

```
import { defineConfig } from '@playwright/test'; export default defineConfig({ webServer: { command:
'npm run build && npm run preview', port: 4173 }, testDir: 'e2e' });
```

postcss.config.js

```
// File: postcss.config.js (CORRIGIDO) export default { plugins: { '@tailwindcss/postcss': {}, //
Usando o novo pacote autoprefixer: {}, }, }
```

prisma\dev.db

```
SQLite format 3■■■@ !6!■■!.v■■■(■■ T ■ '■
■■i■1■p■■■■■e■x■(■(;■■■O)■indexsqlite_autoindex_EmotionCatalog_1EmotionCatalog■■■■■
■otableIndicatorIndicator■CREATE TABLE "Indicator" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT
NULL, "name" TEXT NOT NULL, CONSTRAINT "Indicator_userId_fkey" FOREIGN KEY ("userId") REFERENCES
"User" ("id") ON DELETE CASCADE ON UPDATE CASCADE )1■■■E
■indexsqlite_autoindex_Indicator_1Indicator■■■■■ ■otableTimeframeTimeframe■CREATE TABLE "Timeframe"
( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL, CONSTRAINT
"Timeframe_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE
CASCADE )1■■■E ■indexsqlite_autoindex_Timeframe_1Timeframe■ ■■■ ■ktableModalityModality■CREATE TABLE
"Modality" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL, CONSTRAINT
"Modality_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE
CASCADE )/■■■C ■indexsqlite_autoindex_Modality_1Modality■<■■■ ■GtableAssetTypeAssetType■CREATE TABLE
"AssetType" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL, "currency"
TEXT NOT NULL DEFAULT 'BRL', CONSTRAINT "AssetType_userId_fkey" FOREIGN KEY ("userId") REFERENCES
"User" ("id") ON DELETE CASCADE ON UPDATE CASCADE )1■■■E
■indexsqlite_autoindex_AssetType_1AssetType■C ■■■■■etableAssetAsset■CREATE TABLE "Asset" ( "id" TEXT
NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "ticker" TEXT NOT NULL, "description" TEXT,
"assetTypeId" TEXT NOT NULL, CONSTRAINT "Asset_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User"
("id") ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT "Asset_assetTypeId_fkey" FOREIGN KEY
("assetTypeId") REFERENCES "AssetType" ("id") ON DELETE RESTRICT ON UPDATE CASCADE
))■■■=■■indexsqlite_autoindex_Asset_1Asset■F ■■■■■stableFeeFee CREATE TABLE "Fee" ( "id" TEXT NOT
NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL, "value" REAL NOT NULL, "isPercentage"
BOOLEAN NOT NULL DEFAULT false, CONSTRAINT "Fee_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User"
("id") ON DELETE CASCADE ON UPDATE CASCADE )% ■■9■■indexsqlite_autoindex_Fee_1Fee
■■■■■■tableApiKeyApiKey CREATE TABLE "ApiKey" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT
NULL, "service" TEXT NOT NULL, "key" TEXT NOT NULL, CONSTRAINT "ApiKey_userId_fkey" FOREIGN KEY
("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE CASCADE )+
■■?■■indexsqlite_autoindex_ApiKey_1ApiKey X■■■■■■■tableAccountAccount■CREATE TABLE "Account" ( "id"
TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL, "initialBalance" REAL NOT
NULL, "currency" TEXT NOT NULL DEFAULT 'BRL', CONSTRAINT "Account_userId_fkey" FOREIGN KEY ("userId")
REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE CASCADE
)-■■■A■■indexsqlite_autoindex_Account_1Account y■■■■■■YtableKeyKey■CREATE TABLE "Key" ( "id" TEXT
NOT NULL PRIMARY KEY, "hashedPassword" TEXT, "userId" TEXT NOT NULL, CONSTRAINT "Key_userId_fkey"
FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE CASCADE
)%■■■9■■indexsqlite_autoindex_Key_1Key■■■■■■■■ytableSessionSession■CREATE TABLE "Session" ( "id"
TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "expiresAt" DATETIME NOT NULL, CONSTRAINT
```

```
"Session_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE
CASCADE )-■■■A■■indexsqlite_autoindex_Session_1Session■■■■■■■■stableUserUser■CREATE TABLE "User" (
"id" TEXT NOT NULL PRIMARY KEY, "username" TEXT NOT NULL, "email" TEXT NOT NULL
)'■■■;■■indexsqlite_autoindex_User_1Use■ ■■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
■■■■D■■■■■ T ■ '■
■■i■d■1■"■■■■■e■(■(;■■■O)■indexsqlite_autoindex_EmotionCatalog_1EmotionCatalog■1■■■E
■indexsqlite_autoindex_Indicator_1Indicator■■■■■ ■otableIndicatorIndicator■CREATE TABLE "Indicator"
( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL, CONSTRAINT
"Indicator_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE
CASCADE )1■■■E ■indexsqlite_autoindex_Timeframe_1Timeframe■■■■■ ■otableTimeframeTimeframe■CREATE
TABLE "Timeframe" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL,
CONSTRAINT "Timeframe_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE
ON UPDATE CASCADE )/■■■C ■indexsqlite_autoindex_Modality_1Modality■ ■■■
■ktableModalityModality■CREATE TABLE "Modality" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT
NULL, "name" TEXT NOT NULL, CONSTRAINT "Modality_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User"
("id") ON DELETE CASCADE ON UPDATE CASCADE )1■■■E ■indexsqlite_autoindex_AssetType_1AssetType■<■■■
■GtableAssetTypeAssetType■CREATE TABLE "AssetType" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT
NULL, "name" TEXT NOT NULL, "currency" TEXT NOT NULL DEFAULT 'BRL', CONSTRAINT "AssetType_userId_fkey"
FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE CASCADE )C
■■■■■etableAssetAsset■CREATE TABLE "Asset" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL,
"ticker" TEXT NOT NULL, "description" TEXT, "assetTypeId" TEXT NOT NULL, CONSTRAINT
"Asset_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE
CASCADE, CONSTRAINT "Asset_assetTypeId_fkey" FOREIGN KEY ("assetTypeId") REFERENCES "AssetType" ("id")
ON DELETE RESTRICT ON UPDATE CASCADE ))■■■=■■indexsqlite_autoindex_Asset_1Asset■F ■■■■■stableFeeFee
CREATE TABLE "Fee" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL,
"value" REAL NOT NULL, "isPercentage" BOOLEAN NOT NULL DEFAULT false, CONSTRAINT "Fee_userId_fkey"
FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE CASCADE )%
■■9■■indexsqlite_autoindex_Fee_1Fee ■■■■■■tableApiKeyApiKey CREATE TABLE "ApiKey" ( "id" TEXT NOT
NULL PRIMARY KEY, "userId" TEXT NOT NULL, "service" TEXT NOT NULL, "key" TEXT NOT NULL, CONSTRAINT
"ApiKey_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE
CASCADE )+ ■■?■■indexsqlite_autoindex_ApiKey_1ApiKey X■■■■■■■tableAccountAccount■CREATE TABLE
"Account" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL,
"initialBalance" REAL NOT NULL, "currency" TEXT NOT NULL DEFAULT 'BRL', CONSTRAINT
"Account_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE
CASCADE )-■■■A■■indexsqlite_autoindex_Account_1Account y■■■■■■YtableKeyKey■CREATE TABLE "Key" (
"id" TEXT NOT NULL PRIMARY KEY, "hashedPassword" TEXT, "userId" TEXT NOT NULL, CONSTRAINT
"Key_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE CASCADE
)%■■■9■■indexsqlite_autoindex_Key_1Key■■■■■■■■ytableSessionSession■CREATE TABLE "Session" ( "id"
TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "expiresAt" DATETIME NOT NULL, CONSTRAINT
"Session_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE
CASCADE )-■■■A■■indexsqlite_autoindex_Session_1Session■■■■■■■■stableUserUser■CREATE TABLE "User" (
"id" TEXT NOT NULL PRIMARY KEY, "username" TEXT NOT NULL, "email" TEXT NOT NULL
)'■■■;■■indexsqlite_autoindex_User_1User■■ ■T ■■■■ x ;■ ■■■■■ ■|3g#■■''■
table_TradeToError_TradeToError&CREATE TABLE "_TradeToError" ( "A" TEXT NOT NULL, "B" TEXT NOT NULL,
CONSTRAINT "_TradeToError_A_fkey" FOREIGN KEY ("A") REFERENCES "Trade" ("id") ON DELETE CASCADE ON
UPDATE CASCADE, CONSTRAINT "_TradeToError_B_fkey" FOREIGN KEY ("B") REFERENCES "TradeError" ("id") ON
DELETE CASCADE ON UPDATE CASCADE )4 ■■■■■GtableTradeTrade CREATE TABLE "Trade" ( "id" TEXT NOT NULL
PRIMARY KEY, "userId" TEXT NOT NULL, "entryDate" DATETIME NOT NULL, "exitDate" DATETIME, "type" TEXT
NOT NULL, "quantity" REAL NOT NULL, "entryPrice" REAL NOT NULL, "exitPrice" REAL, "stopLoss" REAL,
"takeProfit" REAL, "netResult" REAL, "context" TEXT, "notes" TEXT, "planAdherence" INTEGER,
"lessonsLearned" TEXT, "createdAt" DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP, "updatedAt" DATETIME
NOT NULL, "accountId" TEXT NOT NULL, "assetId" TEXT NOT NULL, "modalityId" TEXT NOT NULL,
"timeframeId" TEXT NOT NULL, "strategyId" TEXT, CONSTRAINT "Trade_userId_fkey" FOREIGN KEY ("userId")
REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT "Trade_accountId_fkey"
FOREIGN KEY ("accountId") REFERENCES "Account" ("id") ON DELETE RESTRICT ON UPDATE CASCADE, CONSTRAINT
"Trade_assetId_fkey" FOREIGN KEY ("assetId") REFERENCES "Asset" ("id") ON DELETE RESTRICT ON UPDATE
CASCADE, CONSTRAINT "Trade_modalityId_fkey" FOREIGN KEY ("modalityId") REFERENCES "Modality" ("id") ON
DELETE RESTRICT ON UPDATE CASCADE, CONSTRAINT "Trade_timeframeId_fkey" FOREIGN KEY ("timeframeId")
REFERENCES "Timeframe" ("id") ON DELETE RESTRICT ON UPDATE CASCADE, CONSTRAINT "Trade_strategyId_fkey"
FOREIGN KEY ("strategyId") REFERENCES "Strategy" ("id") ON DELETE SET NULL ON UPDATE CASCADE )
+=■■indexsqlite_autoindex_Trade_1Trade!$■■■ ■■tableStrategyStrategy CREATE TABLE "Strategy" ( "id"
TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL, "description" TEXT,
CONSTRAINT "Strategy_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON
UPDATE CASCADE )/ ■■C ■indexsqlite_autoindex_Strategy_1Strategy ,■■■!!■#tableTradeErrorTradeError
CREATE TABLE "TradeError" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT
NULL, "description" TEXT, CONSTRAINT "TradeError_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User"
("id") ON DELETE CASCADE ON UPDATE CASCADE )3■■■G!■indexsqlite_autoindex_TradeError_1TradeError
;■■■O)■indexsqlite_autoindex_EmotionCatalog_1EmotionCatalog■X■■■))■ktableEmotionCatalogEmotionCatalog■CREATE
TABLE "EmotionCatalog" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "name" TEXT NOT NULL,
"impact" TEXT NOT NULL, "description" TEXT, CONSTRAINT "EmotionCatalog_userId_fkey" FOREIGN KEY
("userId") REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE CASCADE
)■■!!■■tableEmotionLogEmotionLog$CREATE TABLE "EmotionLog" ( "id" TEXT NOT NULL PRIMARY KEY, "userId"
TEXT NOT NULL, "date" DATETIME NOT NULL, "intensity" INTEGER NOT NULL, "triggers" TEXT, "tradeId"
TEXT, "emotionCatalogId" TEXT NOT NULL, CONSTRAINT "EmotionLog_userId_fkey" FOREIGN KEY ("userId")
REFERENCES "User" ("id") ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT "EmotionLog_tradeId_fkey"
FOREIGN KEY ("tradeId") REFERENCES "Trade" ("id") ON DELETE SET NULL ON UPDATE CASCADE, CONSTRAINT
"EmotionLog_emotionCatalogId_fkey" FOREIGN KEY ("emotionCatalogId") REFERENCES "EmotionCatalog" ("id")
ON DELETE RESTRICT ON UPDATE CASCADE )■G!■ind1■■■E ■indexsqlite_autoindex_Indicator_1Indicator■■■■■
■otableIndicatorIndicator■CREATE TABLE "Indicator" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT
NULL, "name" TEXT NOT NULL, CONSTRAINT "Indicator_userId_fkey" FOREIGN KEY ("userId") REFERENCES
"User" ("id") ON DELETE CASCADE ON UPDATE CASCADE )1■■■E ■indexsqlite_autoindex_Timeframe_1Timeframe■
■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■■■■■ >
9■■■■■■■■■■■■■■■■■■k2■■7'■■index_TradeToError_B_index_TradeToError6CREATE INDEX
"_TradeToError_B_index" ON
"_TradeToError"("B"){1■■;'■!index_TradeToError_AB_unique_TradeToError5CREATE UNIQUE INDEX
"_TradeToError_AB_unique" ON "_TradeToError"("A", "B"){0■■=
■)indexStrategy_userId_name_keyStrategy4CREATE UNIQUE INDEX "Strategy_userId_name_key" ON
"Strategy"("userId", "name")■/■■A!■1indexTradeError_userId_name_keyTradeError3CREATE UNIQUE INDEX
"TradeError_userId_name_key" ON "TradeError"("userId",
"name")■.■■I)■AindexEmotionCatalog_userId_name_keyEmotionCatalog2CREATE UNIQUE INDEX
"EmotionCatalog_userId_name_key" ON "EmotionCatalog"("userId", "name")■-■■?
■-indexIndicator_userId_name_keyIndicator1CREATE UNIQUE INDEX "Indicator_userId_name_key" ON
"Indicator"("userId", "name")■,■■? ■-indexTimeframe_userId_name_keyTimeframe0CREATE UNIQUE INDEX
"Timeframe_userId_name_key" ON "Timeframe"("userId", "name"){+■■=
■)indexModality_userId_name_keyModality/CREATE UNIQUE INDEX "Modality_userId_name_key" ON
"Modality"("userId", "name")■*■■? ■-indexAssetType_userId_name_keyAssetType.CREATE UNIQUE INDEX
"AssetType_userId_name_key" ON "AssetType"("userId",
"name")u)■■;■■%indexAsset_userId_ticker_keyAsset-CREATE UNIQUE INDEX "Asset_userId_ticker_key" ON
"Asset"("userId", "ticker")M(■■)■■mindexFee_userId_idxFee,CREATE INDEX "Fee_userId_idx" ON
"Fee"("userId")Y'■■/■■yindexApiKey_userId_idxApiKey+CREATE INDEX "ApiKey_userId_idx" ON
"ApiKey"("userId")]&■■1■■}indexAccount_userId_idxAccount*CREATE INDEX "Account_userId_idx" ON
"Account"("userId")M%■■)■■mindexKey_userId_idxKey)CREATE INDEX "Key_userId_idx" ON
"Key"("userId")U$■■)■■{indexUser_email_keyUser'CREATE UNIQUE INDEX "User_email_key" ON
"User"("email")g#■■''■ table_TradeToError_TradeToError&CREATE TABLE "_TradeToError" ( "A" TEXT NOT
NULL, "B" TEXT NOT NULL, CONSTRAINT "_TradeToError_A_fkey" FOREIGN KEY ("A") REFERENCES "Trade" ("id")
ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT "_TradeToError_B_fkey" FOREIGN KEY ("B") REFERENCES
"TradeError" ("id") ON DELETE CASCADE ON UPDATE CASCADE
)3"■■G!■indexsqlite_autoindex_EmotionLog_1EmotionLog%&!■■!!■■tableEmotionLogEmotionLog$CREATE TABLE
"EmotionLog" ( "id" TEXT NOT NULL PRIMARY KEY, "userId" TEXT NOT NULL, "date" DATETIME NOT NULL,
"intensity" INTEGER NOT NULL, "triggers" TEXT, "tradeId" TEXT, "emotionCatalogId" TEXT NOT NULL,
CONSTRAINT "EmotionLog_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User" ("id") ON DELETE CASCADE
ON UPDATE CASCADE, CONSTRAINT "EmotionLog_tradeId_fkey" FOREIGN KEY ("tradeId") REFERENCES "Trade"
("id") ON DELETE SET NULL ON UPDATE CASCADE, CONSTRAINT "EmotionLog_emotionCatalogId_fkey" FOREIGN KEY
("emotionCatalogId") REFERENCES "EmotionCatalog" ("id") ON DELETE RESTRICT ON UPDATE CASCADE )3
■■G!■indexsqlite_autoindex_TradeImage_1TradeImage#■ ■■!!■ytableTradeImageTradeImage"CREATE TABLE
"TradeImage" ( "id" TEXT NOT NULL PRIMARY KEY, "url" TEXT NOT NULL, "tradeId" TEXT NOT NULL,
CONSTRAINT "TradeImage_tradeId_fkey" FOREIGN KEY ("tradeId") REFERENCES "Trade" ("id") ON DELETE
CASCADE ON UPDATE CASCADE )) ■■=■■indexsqlite_autoindex_Trade_1Trade! ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
```

prisma\schema.prisma

```
generator client { provider = "prisma-client-js" } datasource db { provider = "sqlite" url =
"file:./dev.db" } model User { id String @id username String email String @unique sessions Session[]
keys Key[] accounts Account[] apiKeys ApiKey[] fees Fee[] assets Asset[] assetTypes AssetType[]
modalities Modality[] timeframes Timeframe[] indicators Indicator[] emotionCatalogs EmotionCatalog[]
tradeErrors TradeError[] strategies Strategy[] trades Trade[] emotionLogs EmotionLog[] } model Session
{ id String @id userId String expiresAt DateTime user User @relation(references: [id], fields:
[userId], onDelete: Cascade) } model Key { id String @id hashedPassword String? userId String user
User @relation(references: [id], fields: [userId], onDelete: Cascade) @@index([userId]) } model
Account { id String @id @default(cuid()) userId String name String initialBalance Float currency
String @default("BRL") user User @relation(fields: [userId], references: [id], onDelete: Cascade)
trades Trade[] @@index([userId]) } model ApiKey { id String @id @default(cuid()) userId String service
String key String user User @relation(fields: [userId], references: [id], onDelete: Cascade)
@@index([userId]) } model Fee { id String @id @default(cuid()) userId String name String value Float
isPercentage Boolean @default(false) user User @relation(fields: [userId], references: [id], onDelete:
Cascade) @@index([userId]) } model Asset { id String @id @default(cuid()) userId String ticker String
description String? assetTypeId String user User @relation(fields: [userId], references: [id],
onDelete: Cascade) assetType AssetType @relation(fields: [assetTypeId], references: [id]) trades
Trade[] @@unique([userId, ticker]) } model AssetType { id String @id @default(cuid()) userId String
name String currency String @default("BRL") user User @relation(fields: [userId], references: [id],
onDelete: Cascade) assets Asset[] @@unique([userId, name]) } model Modality { id String @id
@default(cuid()) userId String name String user User @relation(fields: [userId], references: [id],
onDelete: Cascade) trades Trade[] @@unique([userId, name]) } model Timeframe { id String @id
@default(cuid()) userId String name String user User @relation(fields: [userId], references: [id],
onDelete: Cascade) trades Trade[] @@unique([userId, name]) } model Indicator { id String @id
@default(cuid()) userId String name String user User @relation(fields: [userId], references: [id],
onDelete: Cascade) @@unique([userId, name]) } model EmotionCatalog { id String @id @default(cuid())
userId String name String impact String description String? user User @relation(fields: [userId],
references: [id], onDelete: Cascade) emotionLogs EmotionLog[] @@unique([userId, name]) } model
TradeError { id String @id @default(cuid()) userId String name String description String? user User
@relation(fields: [userId], references: [id], onDelete: Cascade) trades Trade[]
@relation("TradeToError") @@unique([userId, name]) } model Strategy { id String @id @default(cuid())
userId String name String description String? user User @relation(fields: [userId], references: [id],
onDelete: Cascade) trades Trade[] @@unique([userId, name]) } model Trade { id String @id
@default(cuid()) userId String entryDate DateTime exitDate DateTime? type String quantity Float
entryPrice Float exitPrice Float? stopLoss Float? takeProfit Float? netResult Float? context String?
notes String? planAdherence Int? lessonsLearned String? createdAt DateTime @default(now()) updatedAt
DateTime @updatedAt user User @relation(fields: [userId], references: [id], onDelete: Cascade)
accountId String account Account @relation(fields: [accountId], references: [id]) assetId String asset
Asset @relation(fields: [assetId], references: [id]) modalityId String modality Modality
@relation(fields: [modalityId], references: [id]) timeframeId String timeframe Timeframe
@relation(fields: [timeframeId], references: [id]) strategyId String? strategy Strategy?
@relation(fields: [strategyId], references: [id]) errors TradeError[] @relation("TradeToError") images
TradeImage[] emotionLogs EmotionLog[] } model TradeImage { id String @id @default(cuid()) url String
tradeId String trade Trade @relation(fields: [tradeId], references: [id], onDelete: Cascade) } model
EmotionLog { id String @id @default(cuid()) userId String date DateTime intensity Int triggers String?
tradeId String? emotionCatalogId String user User @relation(fields: [userId], references: [id],
onDelete: Cascade) trade Trade? @relation(fields: [tradeId], references: [id]) emotionCatalog
EmotionCatalog @relation(fields: [emotionCatalogId], references: [id]) }
```

README.md

```
# sv Everything you need to build a Svelte project, powered by
[`sv`](https://github.com/sveltejs/cli). ## Creating a project If you're seeing this, you've probably
already done this step. Congrats! ```bash # create a new project in the current directory npx sv
create # create a new project in my-app npx sv create my-app
```

Developing

Once you've created a project and installed dependencies with `npm install` (or `pnpm install` or `yarn`), start a development server:

```
npm run dev # or start the server and open the app in a new browser tab npm run dev -- --open
```

Building

To create a production version of your app:

```
npm run build
```

You can preview the production build with `npm run preview`.

> To deploy your app, you may need to install an [adapter](#) for your target environment.

```
### `src\app.css` ```css /* File: src/app.css */ @tailwind base; @tailwind components; @tailwind
utilities;
```

`src\app.d.ts`

```
// File: src/app.d.ts declare global { namespace App { // interface Error {} interface Locals { user:
import('lucia').User | null; session: import('lucia').Session | null; } // interface PageData {} //
interface Platform {} } } export {};
```

`src\app.html`

```
<!doctype html> <html lang="en"> <head> <meta charset="utf-8" /> <link rel="icon"
href="%sveltekit.assets%/favicon.svg" /> <meta name="viewport" content="width=device-width,
initial-scale=1" /> %sveltekit.head% </head> <body data-sveltekit-preload-data="hover"> <div
style="display: contents">%sveltekit.body%</div> </body> </html>
```

`src\demo.spec.ts`

```
import { describe, it, expect } from 'vitest'; describe('sum test', () => { it('adds 1 + 2 to equal
3', () => { expect(1 + 2).toBe(3); }); });
```

`src\hooks.server.ts`

```
// File: src/hooks.server.ts import { lucia } from '$lib/server/auth'; import type { Handle } from
'@sveltejs/kit'; export const handle: Handle = async ({ event, resolve }) => { const sessionId =
event.cookies.get(lucia.sessionCookieName); if (!sessionId) { event.locals.user = null;
event.locals.session = null; return resolve(event); } const { session, user } = await
lucia.validateSession(sessionId); if (session && session.fresh) { const sessionCookie =
lucia.createSessionCookie(session.id); event.cookies.set(sessionCookie.name, sessionCookie.value, {
path: '.', ...sessionCookie.attributes }); } if (!session) { const sessionCookie =
lucia.createBlankSessionCookie(); event.cookies.set(sessionCookie.name, sessionCookie.value, { path:
'.', ...sessionCookie.attributes }); } event.locals.user = user; event.locals.session = session;
```

```
return resolve(event); };
```

src\lib\index.ts

```
// place files you want to import through the `$lib` alias in this folder.
```

src\lib\schemas\authSchema.ts

```
// File: src/lib/schemas/authSchema.ts import { z } from 'zod'; export const registerSchema = z
.object({ username: z .string({ required_error: 'Username is required' }) .min(3, { message: 'Username
must be at least 3 characters long' }) .max(20, { message: 'Username must be at most 20 characters
long' }) .regex(/^[a-zA-Z0-9_]+$/, { message: 'Username can only contain letters, numbers, and
underscores' }), email: z .string({ required_error: 'Email is required' }) .email({ message: 'Invalid
email address' }), password: z .string({ required_error: 'Password is required' }) .min(8, { message:
'Password must be at least 8 characters long' }) .max(100, { message: 'Password must be at most 100
characters long' }), passwordConfirm: z .string({ required_error: 'Password confirmation is required'
}) }) .refine((data) => data.password === data.passwordConfirm, { message: 'Passwords do not match',
path: ['passwordConfirm'] // a que campo o erro se aplica }); export const loginSchema = z.object({
email: z .string({ required_error: 'Email is required' }) .email({ message: 'Invalid email address'
}), password: z .string({ required_error: 'Password is required' }) .min(1, { message: 'Password is
required' }) });
```

src\lib\server\auth.ts

```
// File: src/lib/server/auth.ts import { Lucia } from 'lucia'; import { dev } from '$app/environment';
import { PrismaAdapter } from '@lucia-auth/adapter-prisma'; import prisma from './database'; const
adapter = new PrismaAdapter(prisma.session, prisma.user); export const lucia = new Lucia(adapter, {
sessionCookie: { attributes: { // use secure cookies in production secure: !dev } },
getUserAttributes: (attributes) => { return { userId: attributes.id, email: attributes.email,
username: attributes.username }; } }); declare module 'lucia' { interface Register { Lucia: typeof
lucia; DatabaseUserAttributes: { id: string; email: string; username: string; }; } }
```

src\lib\server\database.ts

```
// File: src/lib/server/database.ts import { PrismaClient } from '@prisma/client'; const prisma = new
PrismaClient(); export default prisma;
```

src\routes\(auth)\+layout.svelte

```
<!-- File: src/routes/(auth)/+layout.svelte --> <main class="flex items-center justify-center
min-h-screen"> <div class="card w-full max-w-md bg-base-200 shadow-xl"> <div class="card-body"> <slot
/> </div> </div> </main>
```

src\routes\(auth)\login\+page.server.ts

```
// File: src/routes/(auth)/login/+page.server.ts import { loginSchema } from
'$lib/schemas/authSchema'; import { superValidate } from 'sveltekit-superforms/server'; import { zod }
```

```
from 'sveltekit-superforms/adapters'; import { fail, redirect } from '@sveltejs/kit'; import { lucia }
from '$lib/server/auth'; import { Argon2id } from 'oslo/password'; import prisma from
'$lib/server/database'; export const load = async (event) => { if (event.locals.user) { redirect(302,
'/'); // Redireciona usuários já logados } const form = await superValidate(zod(loginSchema)); return
{ form }; }; export const actions = { default: async (event) => { const form = await
superValidate(event, zod(loginSchema)); if (!form.valid) { return fail(400, { form }); } const {
email, password } = form.data; const existingUser = await prisma.user.findUnique({ where: { email },
include: { keys: true } // Inclui as chaves para verificar a senha }); if (!existingUser) { return
fail(400, { message: 'Incorrect email or password', form }); } const key = existingUser.keys.find(k =>
k.id === 'email'); if (!key || !key.hashedPassword) { // Caso de um usuário que não tenha uma chave de
senha (ex: OAuth no futuro) return fail(400, { message: 'Incorrect email or password', form }); }
const validPassword = await new Argon2id().verify(key.hashedPassword, password); if (!validPassword) {
return fail(400, { message: 'Incorrect email or password', form }); } // Sucesso! Cria a sessão const
session = await lucia.createSession(existingUser.id, {}); const sessionCookie =
lucia.createSessionCookie(session.id); event.cookies.set(sessionCookie.name, sessionCookie.value, {
path: '.', ...sessionCookie.attributes }); redirect(302, '/'); } };
```

src\routes\(auth)\login\+page.svelte

```
<!-- File: src/routes/(auth)/login/+page.svelte --> <script lang="ts"> import { superForm } from
'sveltekit-superforms'; import { zodClient } from 'sveltekit-superforms/adapters'; import {
loginSchema } from '$lib/schemas/authSchema'; import SuperDebug from
'sveltekit-superforms/client/SuperDebug.svelte'; export let data; export let form; // Vem da action em
caso de falha const { form: formData, errors, enhance } = superForm(form ?? data.form, { validators:
zodClient(loginSchema) }); </script> <h1 class="card-title text-2xl mb-4">Login</h1> <!-- <SuperDebug
data={$formData} /> --> {#if form?.message} <div class="alert alert-error mb-4">
<span>{form.message}</span> </div> {/if} <form method="POST" use:enhance> <div class="form-control">
<label for="email" class="label">Email</label> <input type="email" id="email" name="email"
class="input input-bordered" bind:value={$formData.email} aria-invalid={$errors.email ? 'true' :
undefined} /> {#if $errors.email}<span class="text-error text-sm mt-1">{$errors.email}</span>{/if}
</div> <div class="form-control mt-4"> <label for="password" class="label">Senha</label> <input
type="password" id="password" name="password" class="input input-bordered"
bind:value={$formData.password} aria-invalid={$errors.password ? 'true' : undefined} /> {#if
$errors.password}<span class="text-error text-sm mt-1">{$errors.password}</span>{/if} </div> <div
class="card-actions mt-6"> <button type="submit" class="btn btn-primary w-full">Entrar</button> </div>
</form> <div class="text-center mt-4"> <a href="/register" class="link">Não tem uma conta? Crie uma
agora</a> </div>
```

src\routes\(auth)\register\+page.server.ts

```
// File: src/routes/(auth)/register/+page.server.ts import { registerSchema } from
'$lib/schemas/authSchema'; import { superValidate } from 'sveltekit-superforms/server'; import { zod }
from 'sveltekit-superforms/adapters'; import { fail, redirect } from '@sveltejs/kit'; import { lucia }
from '$lib/server/auth'; import { Argon2id } from 'oslo/password'; import prisma from
'$lib/server/database'; export const load = async (event) => { if (event.locals.user) { redirect(302,
'/'); // Redireciona usuários logados } const form = await superValidate(zod(registerSchema)); return
{ form }; }; export const actions = { default: async (event) => { const form = await
superValidate(event, zod(registerSchema)); if (!form.valid) { return fail(400, { form }); } const {
email, username, password } = form.data; // Checa se o usuário já existe const existingUser = await
prisma.user.findFirst({ where: { OR: [{ email }, { username }] } }); if (existingUser) { return
fail(400, { message: 'A user with that email or username already exists.', form }); } const
hashedPassword = await new Argon2id().hash(password); const userId = crypto.randomUUID(); // Gera um
ID único await prisma.user.create({ data: { id: userId, username, email, keys: { create: { id:
'email', // providerId para Lucia hashedPassword } } } }); const session = await
lucia.createSession(userId, {}); const sessionCookie = lucia.createSessionCookie(session.id);
event.cookies.set(sessionCookie.name, sessionCookie.value, { path: '.', ...sessionCookie.attributes
}); redirect(302, '/'); } };
```

src\routes\(auth)\register\+page.svelte

```
<!-- File: src/routes/(auth)/register/+page.svelte --> <script lang="ts"> import { superForm } from
'sveltekit-superforms'; import { zodClient } from 'sveltekit-superforms/adapters'; import {
registerSchema } from '$lib/schemas/authSchema'; import SuperDebug from
'sveltekit-superforms/client/SuperDebug.svelte'; export let data; const { form, errors, enhance } =
superForm(data.form, { validators: zodClient(registerSchema) }); </script> <h1 class="card-title
text-2xl mb-4">Criar Conta</h1> <!-- <SuperDebug data={$form} /> --> <form method="POST" use:enhance>
<div class="form-control"> <label for="username" class="label">Username</label> <input type="text"
id="username" name="username" class="input input-bordered" bind:value={$form.username}
aria-invalid={$errors.username ? 'true' : undefined} /> {#if $errors.username}<span class="text-error
text-sm mt-1">{$errors.username}</span>{/if} </div> <div class="form-control mt-4"> <label for="email"
class="label">Email</label> <input type="email" id="email" name="email" class="input input-bordered"
bind:value={$form.email} aria-invalid={$errors.email ? 'true' : undefined} /> {#if $errors.email}<span
class="text-error text-sm mt-1">{$errors.email}</span>{/if} </div> <div class="form-control mt-4">
<label for="password" class="label">Senha</label> <input type="password" id="password" name="password"
class="input input-bordered" bind:value={$form.password} aria-invalid={$errors.password ? 'true' :
undefined} /> {#if $errors.password}<span class="text-error text-sm
mt-1">{$errors.password}</span>{/if} </div> <div class="form-control mt-4"> <label
for="passwordConfirm" class="label">Confirmar Senha</label> <input type="password"
id="passwordConfirm" name="passwordConfirm" class="input input-bordered"
bind:value={$form.passwordConfirm} aria-invalid={$errors.passwordConfirm ? 'true' : undefined} /> {#if
$errors.passwordConfirm}<span class="text-error text-sm mt-1">{$errors.passwordConfirm}</span>{/if}
</div> <div class="card-actions mt-6"> <button type="submit" class="btn btn-primary
w-full">Registrar</button> </div> </form> <div class="text-center mt-4"> <a href="/login"
class="link">Já tem uma conta? Faça o login</a> </div>
```

src\routes\+layout.svelte

```
<!-- File: src/routes/+layout.svelte (CORRIGIDO) --> <script lang="ts"> import '../app.css'; // O
caminho agora está correto. </script> <div class="min-h-screen bg-base-100 text-base-content
font-sans"> <!-- O SvelteKit renderizará todas as nossas páginas aqui --> <slot /> </div>
```

src\routes\+page.svelte

```
<h1>Welcome to SvelteKit</h1> <p>Visit <a href="https://svelte.dev/docs/kit">svelte.dev/docs/kit</a>
to read the documentation</p>
```

src\routes\page.svelte.test.ts

```
import { page } from '@vitest/browser/context'; import { describe, expect, it } from 'vitest'; import
{ render } from 'vitest-browser-svelte'; import Page from './+page.svelte'; describe('/+page.svelte',
() => { it('should render h1', async () => { render(Page); const heading = page.getByRole('heading', {
level: 1 }); await expect.element(heading).toBeInTheDocument(); }); });
```

svelte.config.js

```
// File: svelte.config.js import adapter from '@sveltejs/adapter-auto'; import { vitePreprocess } from
'@sveltejs/vite-plugin-svelte'; /** @type {import('@sveltejs/kit').Config} */ const config = { //
Consult https://kit.svelte.dev/docs/integrations#preprocessors // for more information about
preprocessors preprocess: vitePreprocess(), // <--- ESTA LINHA É A CHAVE kit: { // adapter-auto only
supports some environments, see https://kit.svelte.dev/docs/adapter-auto for a list. // If your
environment is not supported or you settled on a specific environment, switch out the adapter. // See
https://kit.svelte.dev/docs/adapters for more information about adapters. adapter: adapter() } };
export default config;
```

tailwind.config.js

```
// File: tailwind.config.js (COM A LINHA ADICIONADA) /** @type {import('tailwindcss').Config} */
export default { content: [ './src/**/*.{html,js,svelte,ts}', './src/routes/+layout.svelte' // <--
ADICIONE ESTA LINHA ESPECÍFICA ], theme: { extend: {}, }, plugins: [ require('@tailwindcss/forms'),
require('daisyui') ], daisyui: { themes: [ "light", "dark", "corporate", "dracula", "forest", "lofi",
], }, };
```

tsconfig.json

```
{ "extends": "./.svelte-kit/tsconfig.json", "compilerOptions": { "allowJs": true, "checkJs": true,
"esModuleInterop": true, "forceConsistentCasingInFileNames": true, "resolveJsonModule": true,
"skipLibCheck": true, "sourceMap": true, "strict": true, "moduleResolution": "bundler" } // Path
aliases are handled by https://svelte.dev/docs/kit/configuration#alias // except $lib which is handled
by https://svelte.dev/docs/kit/configuration#files // // If you want to overwrite includes/excludes,
make sure to copy over the relevant includes/excludes // from the referenced tsconfig.json -
TypeScript does not merge them in }
```

vite.config.ts

```
import { sveltekit } from '@sveltejs/kit/vite'; import { defineConfig } from 'vite'; export default
defineConfig({ plugins: [sveltekit()], test: { projects: [ { extends: './vite.config.ts', test: {
name: 'client', environment: 'browser', browser: { enabled: true, provider: 'playwright', instances:
[{ browser: 'chromium' }] }, include: ['src/**/*.svelte.{test,spec}.{js,ts}'], exclude:
['src/lib/server/**'], setupFiles: ['./vitest-setup-client.ts'] } }, { extends: './vite.config.ts',
test: { name: 'server', environment: 'node', include: ['src/**/*.{test,spec}.{js,ts}'], exclude:
['src/**/*.svelte.{test,spec}.{js,ts}'] } } ] } });
```

vitest-setup-client.ts

```
/// <reference types="@vitest/browser/matchers" /> /// <reference
types="@vitest/browser/providers/playwright" />
```