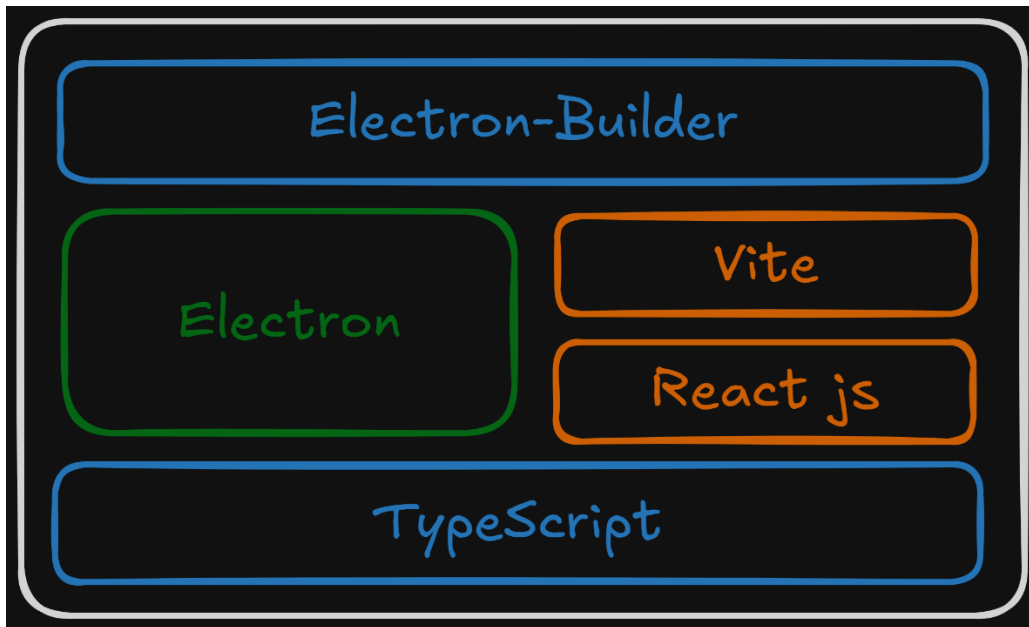


الکترون در ری اکت جی اس (Electron + React js + TypeScript):



موارد مورد استفاده در این ساختار پروژه :

1. TypeScript : در این پروژه برای توسعه از زبان تایپ اسکریپت استفاده شده.
2. Vite: از vite به عنوان یک باندلر به روز برای توسعه و کانفیگ و ارتباط بین تکنولوژی ها و ساختار اصلی پروژه استفاده شده .
3. React js : از این کتابخانه برای توسعه UI پروژه استفاده شده است .
4. Electron : از این فریمورک برای ساخت پروژه اصلی که یک برنامه تحت دسکتاپ به صورت چندسکویی است استفاده می کنیم .
5. Electron-Bulder : در نهایت از این پکیج برای گرفتن خروجی نهایی پروژه برای سیستم عامل های : ویندوز ، مک و لینوکس استفاده می کنیم.

مراحل ایجاد پروژه :

- (1) با اجرای دستور `npm create vite` شروع به نصب vite می کنیم و بعد React و تایپ اسکریپت را به عنوان مقادیر انتخابی برای نصب انتخاب می کنیم .
- (2) بعد از نصب vite به دستور `npm i` فایل نودماژول را نصب می کنیم .
- (3) به فایل `index.html` رفته و مسیر فایل اسکریپت را به `"src/ui/main.tsx"` تغییر می دهیم .
- (4) فایل `main.tsx` و دیگر فایل های پایه مربوط به پروژه react را در پوشه `ui` که در مسیر `src` ایجاد می کنیم قرار می دهیم.
- (5) به فایل `vite.config.ts` میرویم و مقدار `outDir` را با `dist-react` تنظیم می کنیم تا بیلد پروژه ری اکت در این پروژه قرار بگیرد
- (6) در فایل `gitignore` نام `dist-raect` را اضافه می کنیم .
- (7) یک `build` از پروژه react می گیریم .
- (8) با اجرای دستور `npm i -save-dev electron` فریمورک الکترون را به پروژه خود اضافه می کنیم .
- (9) به مسیر `src` باز میگردیم و یک پوشه به نام `electron` می سازیم و فایل `main` مربوط به الکترون را ایجاد می کنیم .
- (10) به فایل `package.json` میرویم و فایل `main` را برای پروژه الکترون خود مشخص می کنیم .
- (11) در قدم بعد اسکریپت های `div:react` و `div:electron` را برای اجرای دو پروژه در زمان توسعه مشخص می کنیم.
- (12) به فایل `vite.config.ts` میرویم و مقدار `base` را با `"/"` مشخص می کنیم تا پروژه در لود فایل های `css` و `js` به مشکل نخورد و مجدد یک بیلد میگیریم
- (13) به فایل `tsconfig.json` می رویم و به عنوان `exclude` مقدار `"src/electron"` را مشخص می کنیم تا کامپایلر تایپ اسکریپت از فایل های

مربوط به پروژه الکترون چشم پوشی کند زیرا میخواهیم کانفیگ جداگانه مشخص کنیم.

(14) به دایرکتوری مربوط به پروژه الکترون می رویم و فایل tsconfig.json جدید را ایجاد می کنیم و کانفیگ های مربوطه را در آن قرار می دهیم و تایپ فایل main را به ts تغییر می دهیم.

```
electron-react > src > electron > tsconfig.json > ...
1  {
2    "compilerOptions": {
3      // require strict types (null-save)
4      "strict": true,
5      // tell TypeScript to generate ESM Syntax
6      "target": "ESNext",
7      // tell TypeScript to require ESM Syntax as input (including .js file imports)
8      "module": "NodeNext",
9      // define where to put generated JS
10     "outDir": "../dist-electron",
11     // ignore errors from dependencies
12     "skipLibCheck": true
13   }
14 }
15
```

(15) به فایل package.json می رویم و اسکریپت مربوط به کامپایل پروژه الکترون از تایپ اسکریپت به جاوااسکریپت را اضافه می کنیم.

```
"transpile:electron": "tsc --project src/electron/tsconfig.json",
```

(16) در فایل gitignore فایل dist-electron را معرفی می کنیم.

(17) با اجرای دستور npm i -save-dev electron-builder پکیج مربوط به بیلد نهایی برای سیستم عامل های مختلف را نصب می کنیم.

(18) در روت پروژه فایلی با نام electron-builder.json برای قرار دادن کانفیگ های مربوط به بیلد نهایی پروژه می سازیم و کدهای مربوطه را در آن قرار میدهیم.

```
electron-react > {} electron-builder.json > ...
1  {
2    "appId": "com.companyName.projectName",
3    "files": ["dist-electron", "dist-react"],
4    "icon": "./desktopIcon.png",
5    "mac": {
6      "target": "dmg"
7    },
8    "linux": {
9      "target": "AppImage",
10     "category": "Utility"
11   },
12   "win": {
13     "target": ["portable", "msi"]
14   }
15 }
```

(19) به فایل package.json میرویم و اسکریپت های مربوط به بیلد گرفتن نهایی برای سیستم عامل های مختلف را مشخص می کنیم.

```
"dist:mac": "npm run transpile:electron && npm run build && electron-builder --mac --arm64",
"dist:win": "npm run transpile:electron && npm run build && electron-builder --win --x64",
"dist:linux": "npm run transpile:electron && npm run build && electron-builder --linux --x64"
```

(20) برای این که در زمان توسعه با هربار تغییر کد لازم نباشد مجدد از پروژه بیلد بگیریم و این باعث بشه روند توسعه کند و طاقت فرسا بشه باید یک ترفند برای بهینه سازی این روند پیاد سازی کنیم.

(21) با اجرای npm i -save-dev cross-env پکیج مربوطه برای خواندن متغیرهای محیطی در هر سیستم عامل را نصب می کنیم.

(22) اسکریپت dev:electron را با مقدار cross-env

. electron NODE_ENV=development تغییر می دهیم.

(23) در مسیر پروژه الکترون یک فایل utils.ts می سازیم و در آن یک تابع برای

چک کردن اینکه آیا در حالت توسعه هستیم یا نه میسازیم.

```
electron-react > src > electron > util.ts > ...
1 export function isDev(): boolean {
2   return process.env.NODE_ENV === 'development';
3 }
4
```

(24) می توانیم در کانفیگ های vite پورت سرور پروژه ری اکت رو به مقدار

دلخواه تغییر دهیم .

```
electron-react > vite.config.ts > default > server > port
1 import { defineConfig } from 'vite';
2 import react from '@vitejs/plugin-react';
3
4 // https://vitejs.dev/config/
5 export default defineConfig({
6   plugins: [react()],
7   base: './',
8   build: {
9     outDir: 'dist-react',
10  },
11  server: {
12    port: 5123,
13    strictPort: true,
14  },
15 });
16
```

(25) در فایل main.ts مربوط به پروژه الکترون مشخص می کنیم تا در صورت اینکه در حالت توسعه بودیم آدرس لوکال را رندر کند و در غیر این صورت سورس نهایی پروژه را از فایل بیلد شده رندر کند.

```
electron-react > src > electron > main.ts > ...
1 import { app, BrowserWindow } from 'electron';
2 import path from 'path';
3 import { isDev } from './util.js';
4
5 type test = string;
6
7 app.on('ready', () => {
8   const mainWindow = new BrowserWindow({});
9   if (isDev()) {
10     mainWindow.loadURL('http://localhost:5123');
11   } else {
12     mainWindow.loadFile(path.join(app.getAppPath(), '/dist-react/index.html'));
13   }
14 });
```

(26) حالا با هر تغییر در کد ظاهر و UI خروجی را هم در لوکال و هم در اپلیکیشن می بینیم.

(27) با اجرای دستور npm i --save-dev npm-run-all میتوانیم پکیج مربوط به اجرای همه دستورات به صورت یکجا را اضافه کنیم. و حالا اسکریپت مربوط به آن را اضافه کنیم و فقط با اجرای یک دستور هم پروژه react و هم پروژه electron را در یک ترمینال اجرا کنیم.