# Web Technology Project Report

## **Group Number:** 3

## **Topic:** Deno

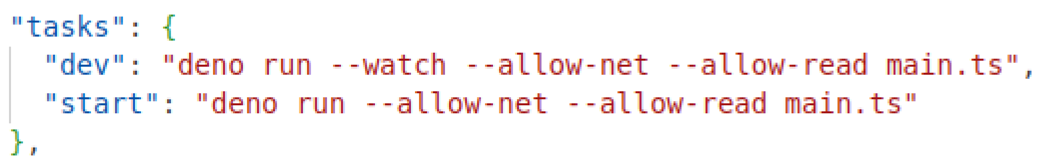
## **Project Name:** Deno Todo API

### Application Description

This project is a RESTful Todo API built with Deno and the Oak framework. We used TypeScript to ensure type safety and an in-memory list to handle CRUD (Create, Read, Update, Delete) operations. The project has a modular structure; we separated routes, types, and the main server code into different files to make it easier to manage.

### 1. Permission System

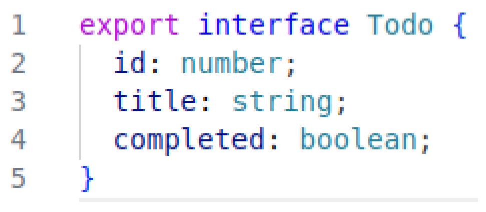
Unlike Node.js, Deno is secure by default. It cannot access the network or the file system unless we explicitly enable it. In our project, we had to use the --allow-net flag to start the server and --allow-read to serve static files.



```
"tasks": {
  "dev": "deno run --watch --allow-net --allow-read main.ts",
  "start": "deno run --allow-net --allow-read main.ts"
},
```

### 2. Native TypeScript Support

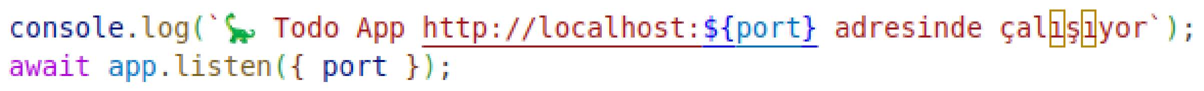
Deno runs TypeScript files directly. We don't need a separate tsconfig.json file or a compile step like in Node.js. In our project, we created a Todo interface to define our data structure. This keeps our data types consistent and prevents errors.



```
export interface Todo {
  id: number;
  title: string;
  completed: boolean;
}
```

### 3. Top-Level Await

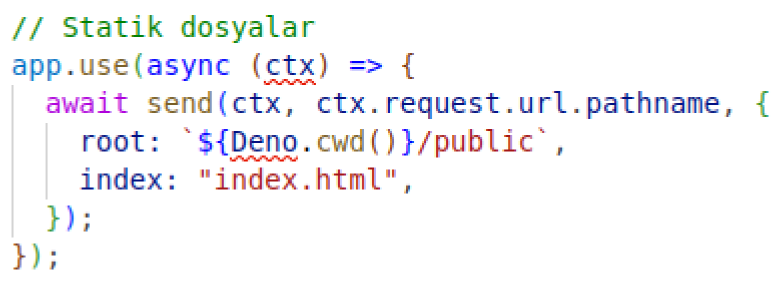
In Deno, we can use the await keyword at the top level of the file. We don't have to wrap it inside an async function. We used this feature in our main.ts file to start the server. It makes the starting code much simpler and cleaner.



```
console.log(`🦕 Todo App http://localhost:${port} adresinde çalışıyor`);
await app.listen({ port });
```

### 4. Deno Runtime API

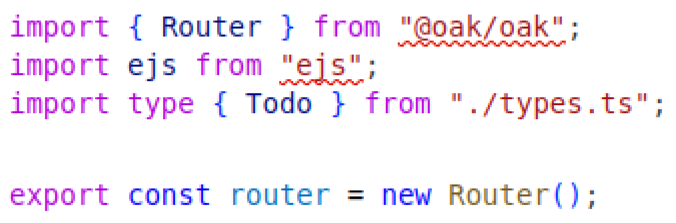
Deno has a global Deno object to access system features securely. We used Deno.cwd() (Current Working Directory) in our code. This helps us find the correct path for our static files folder dynamically.



```
// Statik dosyalar
app.use(async (ctx) => {
  await send(ctx, ctx.request.url.pathname, {
    root: `${Deno.cwd()}/public`,
    index: "index.html",
  });
});
```

### 5. Native ES Modules

The project uses standard ES Modules (import and export). This is the default system in Deno. We created the router in a separate file (routes.ts) and exported it. Then, we imported it into the main file. This makes our code more organized.



```
import { Router } from "@oak/oak";
import ejs from "ejs";
import type { Todo } from "./types.ts";

export const router = new Router();
```