SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Leo Ćavar

IZRADA APLIKACIJE ZA PRONALAZAK TERMINA SASTANAKA

ZAVRŠNI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Leo Ćavar

Matični broj: 0016153823

Studij: Informacijski i poslovni sustavi

IZRADA APLIKACIJE ZA PRONALAZAK TERMINA SASTANAKA

ZAVRŠNI RAD

Mentor :

Doc. Dr. sc. Marko Mijač

Varaždin, Rujan 2024.

Leo Ćavar

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom završnom radu se obrađuje korištenje .NET tehnologija za izradu programa za dogovaranje sastanka. Kroz rad se obrađuje kratko o ASP. NET Core i ASP. NET Web API tehnologijama, kao i koncepte API-ja i HTTP metoda, Google API-ja, uključujući Google Calendar API, autentifikaciju i autorizaciju korisnika pomoću OAuth 2.0 protokola, te primjere zahtjeva za dohvaćanje i upravljanje kalendarskim događajima kroz implementaciju .NET web aplikacije

Ključne riječi: API; ASP. NET Core; .NET; C#; ASP. NET; ; OAuth 2.0; Google Calendar; Google API;

Sadržaj

1. Uvod

U ovom radu ćemo se usredotočiti na izradu programa za dogovaranje sastanaka kroz upotrebu Google API-ja, koji nam omogućava interakciju s iznimno popularnim Google kalendarom. Realizirat ćemo program koristeći ASP.NET Core i Razor stranice za izradu front-end sučelja te upravljanje podacima, dok će ASP.NET Web API služiti za primanje HTTP zahtjeva i komuniciranje s Google servisima za manipuliranje događajima.

2. Metode i tehnike rada

Za pisanje teksta i formatiranje ovog rada koristio se LaTeX unutar programa Visual Studio Code. Za izradu praktičnog dijela korišten je Visual Studio 2022 i JetBrains Rider.

3. API

Kada korisnik koristi softver kao klijent, često koristi nekakvo softversko sučelje za interakciju s softverom, ali kada je potrebno da jedan softver koristi dijelove drugog softvera tada koristimo vrstu sučelja za programiranje aplikacija (engl. *Application Programming Interface*) ili skraćeno API.[1] Ta interakcija se najčešće bazira na tome da klijent šalje HTTP zahtjev serveru na određenu lokaciju i dobivaju se nazad podaci. API zahtjev se sastoji od nekoliko dijelova [2]

- Operacija koja se izvršava (primjer. *GET, POST*)
- Autentifikacijski parametri
- Odredište - URL API završne točke (engl. *endpoint*)

Poziv može sadržavati i druge parametre ali ovo su tri osnovna koja će se uvijek koristiti.

3.1. HTTP zahtjevi

Kada klijent šalje zahtjev poslužitelju mora specifirati u zahtjevu koju metodu želi izvršiti, imena metoda se odnose na ono što želimo postići sa zahtjevom. [3]

- **GET** metoda se koristi za dohvaćanje podataka
- **POST** - slanje i dodavanje podataka
- **PUT** - Ažuriranje podataka
- **DELETE** - brisanje resursa

3.2. RESTFul API

RESTFul API je vrsta API-ja koja prati REST *(*eng. representational state transfer) principe dizajna, može biti u bilo kojem jeziku i može koristiti bilo koju vrstu podataka [4]. Iako najčešće koristi HTTP protokol on nije nužno vezan za njega.[5]

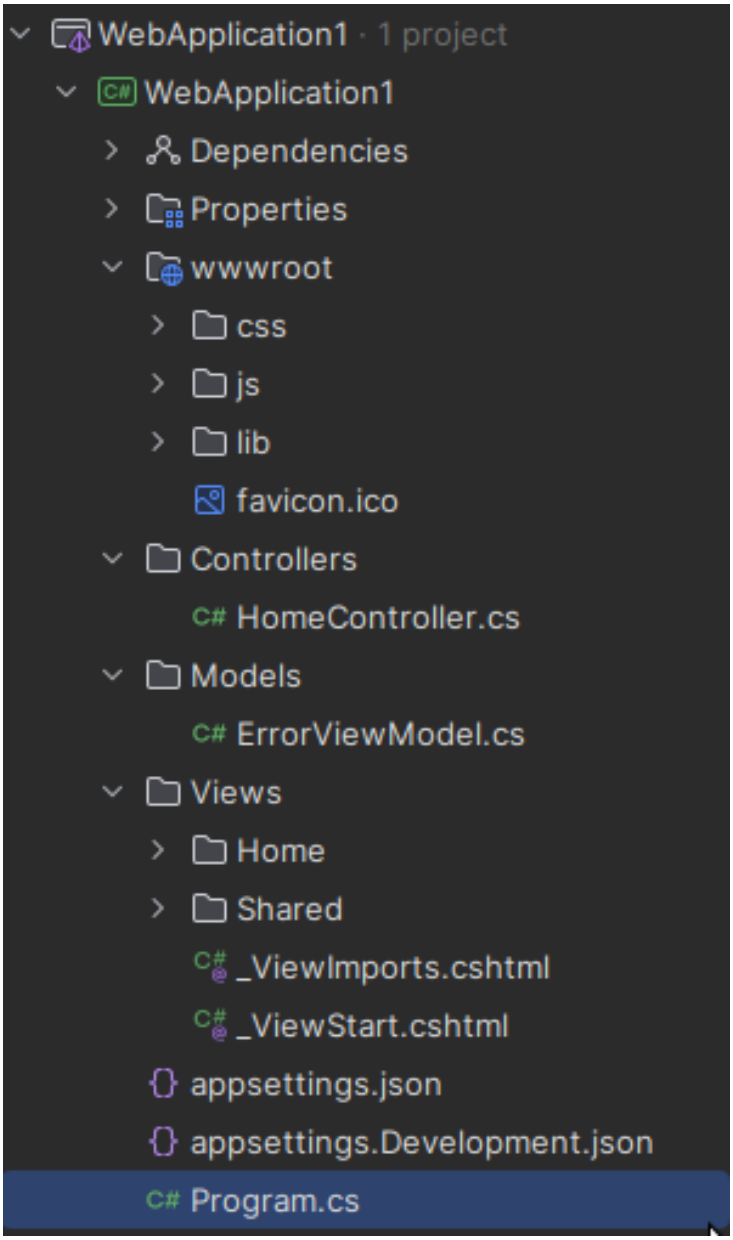
- **Jedinstveno sučelje** - API dizajn mora biti konzistentan i predvidljiv, s pristupom resursima putem standardnih HTTP metoda kao što su GET, POST, PUT i DELETE.
- **Razdvajanje klijenta i servera** - Klijent i server su neovisni, gdje server ne čuva informacije o stanju klijenta između zahtjeva, a klijent nema direktan pristup serverovim podatcima.
- **Bezustanje (engl. *Stateless*)** - Svaki zahtjev od klijenta prema serveru mora sadržavati sve potrebne informacije za obradu, bez potrebe za čuvanjem stanja na serveru.

- **Keširanje** - Resursi se mogu keširati kako bi se smanjilo opterećenje servera i omogućilo ponovnu upotrebu već preuzetih podataka.
- **Sustav slojeva** - Slojevita arhitektura omogućuje umetanje posrednika između klijenta i servera, dodajući funkcionalnosti poput keširanja ili sigurnosnih provjera.
- **Kôd na zahtjev (opcionalno)** - Klijent može preuzeti i izvršiti kod od servera radi proširenja funkcionalnosti aplikacije.

4. ASP.NET Core

4.1. ASP.NET MVC

ASP.NET MVC je framework koji se bazira na MVC (engl. *Model-View-Controller*) arhitekturi, izgrađen je na .NET platformi i koristi se za izradu web aplikacija [6]. Kao što ime glasi, MVC arhitektura se sastoji od modela, pregleda (eng. *View*) i kontrolera (eng. *Controller*). Ovaj oblik dizajna prati prvi princip SOLID metoda, razdvajanja odgovornosti (eng. *Seperation of concerns*). MVC omogućava ponovnu upotrebljivost, i zbog podjele na 3 glavne komponente olakšava održavanje koda. [7]



Slika 1: Prikaz MVC u ASP.NET MVC projektu (Izvor: autor)

4.1.1. Model

Unutar konteksta ASP.NET MVC projekta, model predstavlja C# klasu koja sadrži svojstva za spremanje podataka kojima upravljamo. Model je neovisan o korisničkom sučelju ali često će postojati pregled (engl. textitView) koji odgovara za prikaz i upravljanje modelom. Model također može sadržavati poslovnu logiku za upravljanje podacima, iako to nije učestala praksa i često se ta uloga daje servisima.

4.1.2. View

Pregledi (eng. *Views*) se koriste za prikazivanje podataka i korisničku interakciju. ASP.NET MVC koristi Razor stranice, sa ekstenzijom *cshtml*. Razor stranice omogućavaju pisanje C# koda unutar HTML datoteka koji služi za interaktiranje sa HTML oznakama za generiranje web sadržaja [8]. Najčešće će svaki pregled imati svoj kontroler koji je odgovoran za rad s pregledima.

- **Dijelomični pogledi** (engl. *partial views*) omogućuju smanjenje dupliciranja koda upravljanjem ponovljivim dijelovima pogleda. Primjerice, dijelomični pogled je prikladan za biografiju autora na blogu koja se pojavljuje u više pogleda.
- **Komponente pogleda** (engl. *view components*) slične su dijelomičnim pogledima po tome što smanjuju ponavljanje koda, ali su prikladne za sadržaj pogleda koji zahtijeva izvršavanje koda na poslužitelju za generiranje web stranice.

4.1.3. Controller

4.2. ASP.NET Web API

5. Zaključak

.NET okruženje za skriptiranje u GNU/Linux naredbenom retku pruža nove mogućnosti za razvoj i automatizaciju. Kroz rad je demonstriran veći broj Bash i .NET skripti i demonstrirana je integracija .NET alata i Bash ljuske koristeći .NET alat dotnet-shell. Kroz primjere su prikazane prednosti i nedostatci oba sustava. .NET poboljšava interoperabilnost time što se skripte mogu direktno koristiti na svim operacijskim sustavima koji imaju instalirano .NET okruženje. Također, .NET pruža dodatne funkcionalnosti preko svojih biblioteka i NuGet paketa što omogućuje jednostavno razvijanje kompleksnijih skripti čime se još više mogu poboljšati radni procesi. Prednost Bash skripti je što rade na svim GNU/Linux distribucijama koje koriste Bash ljusku bez potrebe za dodatnim instalacijama i zahtjevaju manje resursa od .NET skripti čime su lakše za sustave. Da zaključim, sinergija .NET okruženja i Bash skriptnog jezika u GNU/Linux operacijskom sustavu omogućuje moćno okruženje za projekte automatizacije iz razloga što se može koristiti najbolje od oba jezika pri pisanju skripti, .NET elementi za kompleksne unaprijed pripremljene funkcionalnosti, a Bash za GNU/Linux specifične radnje ukoliko ima potrebe za njima.

Popis literature

[1] M. Biehl, *API Architecture* (API-University Series). CreateSpace Independent Publishing Platform, 2015., ISBN: 9781508676645. adresa: https://books.google.ba/books?id=6D64DwAAQBAJ.

[2] 3. AltexSoft, *"What is API: Definition, Types, Specifications, Documentation". altexsoft*, https://www.altexsoft.com/blog/what-is-api-definition-types-specifications-documentation/ (Pristupano: 31.7.2024.)

[3] R. Maurya, K. A. Nambiar, P. Babbe, J. P. Kalokhe, Y. S. Ingle i N. F. Shaikh, *Application of Restful APIs in IOT: A Review*, https://www.ijraset.com (Pristupano: 9.8.2024.), 2021.

[4] IBM, *What is a REST API?* https://www.ibm.com/topics/rest-apis (Pristupano: (1.8.2024)).

[5] Microsoft, *RESTful web API design*, https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design (Pristupano: 1.8.2024.), 2023.

[6] C. Tyler, *ASP.NET MVC Tutorial for Beginners: What is, Architecture*, https://example.com (Pristupano: 12.8.2024.), 2024.

[7] GeeksforGeeks, *MVC Design Pattern*, https://www.geeksforgeeks.org/mvc-design-pattern/ (Pristupano: 12.8.2024.), veljača 2024.

[8] S. Smith i D. Brock, *Views in ASP.NET Core MVC*, https://learn.microsoft.com/en-us/aspnet/core/mvc/views/overview?view=aspnetcore-6.0 (Pristupano: 12.8.2024.), 2022.

Popis slika

Popis tablica