**НАЦИОНАЛНА ПРОФЕСИОНАЛНА ТЕХНИЧЕСКА ГИМНАЗИЯ „ШАНДОР ПЕТЬОФИ“**

Разград, ул. Илия Петров № 1, тел. 084/660235, e-mail: [info-1702602@edu.mon.bg](mailto:info-1702602@edu.mon.bg), URL: www.ptgrz.org

**ДИПЛОМЕН ПРОЕКТ**

**за придобиване трета степен на професионална квалификация**

**ТЕМА:**

**Ученик**:

**Професия:**

**Специалност**:

**Ръководител-консултант:**

**Разград**

**2024 година**

Защо Реших да Направя Учебник за Конкурентно Програмиране за Дипломния Проект

Въведение: Конкурентното програмиране е фундаментален аспект на съвременното софтуерно инженерство, позволяващо на приложенията да изпълняват множество операции паралелно, което води до значително подобрение на производителността и ефективността. Тази способност е особено важна в ерата на мултиядрени процесори и разпределени системи, където оптималното използване на ресурсите и бързината на изпълнение са ключови.

Избор на тема: Изборът ми на тема за дипломен проект идва от убеждението, че въпреки значимостта на конкурентното програмиране, образователните ресурси в тази област често са или твърде теоретични, или не достатъчно практически ориентирани. Моята амбиция беше да създам учебник, който да служи като мост между теорията и практиката, като предоставя конкретни примери, упражнения и задачи, които да подготвят читателите за реални програмистки предизвикателства.

Цели на проекта: Главната цел на моя дипломен проект е да предложа учебник, който да обхваща широк спектър от теми в конкурентното програмиране - от основни концепции като нишки и заключвания, до по-сложни теми като модели на конкурентност и асинхронно програмиране. Чрез този учебник, целя да предоставя студентите и професионалистите със солидна основа, която да им позволи да проектират и разработват ефективен многонишков и разпределен софтуер.

Значение и принос: Създаването на този учебник представлява значителен принос към образователните ресурси в областта на програмирането. Той ще позволи на читателите да разберат и приложат конкурентни програмни модели в практиката, като същевременно развият критично мислене и умения за решаване на проблеми в контекста на многонишково програмиране. Освен това, учебникът е предназначен да стимулира интереса и да разшири знанията на читателите за най-новите тенденции и технологии в конкурентното програмиране.

Заключение: Изборът на създаване на учебник за конкурентно програмиране като дипломен проект беше мотивиран от желанието да се адресира нарастващата нужда от качествени образователни материали в тази важна област. С този проект се надявам не само да предоставя ценен ресурс за обучение и самоусъвършенстване, но и да вдъхновя следващото поколение програмисти да продължат да изследват и иновират в областта на конкурентното програмиране

Защо Избрах ASP.NET Core 3.1

Въведение: ASP.NET Core 3.1,

последната стабилна версия преди прехода към .NET 5, представлява водеща технология за разработка на модерни уеб приложения и API-та. Със своята производителност, гъвкавост и поддръжка на крос-платформено разработване, ASP.NET Core 3.1 е идеален избор за разработчици, които търсят надеждна и съвременна основа за своите проекти.

Избор на тема: Изборът на ASP.NET Core 3.1 беше мотивиран от няколко ключови фактора. Първо, производителността и оптимизацията на технологията гарантират, че приложенията, разработени с нея, са бързи и ефективни. Второ, крос-платформената поддръжка позволява на приложенията да се изпълняват на различни операционни системи, което увеличава тяхната достъпност и универсалност. Трето, обширната екосистема и поддръжката на общността предлагат множество ресурси и инструменти за ускоряване на процеса на разработка.

Цели на проекта: Главната цел на моя дипломен проект беше да разработя уеб приложение, което не само да отговори на съвременните изисквания за производителност и безопасност, но и да демонстрира практическото приложение на ASP.NET Core 3.1 в разработката на комплексни уеб решения. Чрез този проект исках да покажа как ASP.NET Core 3.1 може да бъде използван за създаването на високопроизводителни, сигурни и мащабируеми уеб приложения, които могат лесно да се адаптират към изменящите се бизнес изисквания и технологични тенденции. Значение и принос: Изборът на ASP.NET Core 3.1 за моя дипломен проект не е случаен. Тази технология предлага значителни предимства за разработчиците, включително улеснена работа с бази данни, вградена поддръжка за аутентификация и сигурност, както и лесна интеграция с фронтенд фреймуърки като Angular или React. В допълнение, проектът има за цел да демонстрира устойчивостта на ASP.NET Core 3.1 в създаването на API-та, които могат лесно да бъдат консумирани от различни клиентски приложения, подобрявайки така взаимодействието между клиент и сървър. Заключение: Изборът на ASP.NET Core 3.1 за дипломния ми проект беше стратегически решение, основано на желанието да разработя уеб приложение, което не само да отговаря на актуалните технологични стандарти, но и да предостави стабилна платформа за бъдещо развитие и иновации. С този проект се надявам да допринеса към развитието на уеб технологиите и да покажа възможностите, които ASP.NET Core 3.1 предлага за създаването на съвременни и ефективни уеб решения.

Защо Избрах MVC

Решението да използвам MVC модела в моя дипломен проект беше мотивирано от неговата способност да разделя приложението на три основни компонента: модел (Model), изглед (View) и контролер (Controller). Тази разделеност улеснява разработката, тестването и поддръжката на приложението, като в същото време предлага гъвкавост за разширяване и адаптиране към нови изисквания.

Чрез прилагането на MVC архитектурата, целя да постигна висока степен на разделение на отговорностите в кода, което да доведе до по-лесно управление на сложността и по-добра поддръжка на приложението. Това също така улеснява колаборацията между разработчиците, тъй като различните аспекти на приложението (бизнес логика, потребителски интерфейс и контрол на потока на данните) могат да бъдат разработвани паралелно и независимо.

Използването на MVC в моя дипломен проект не само подобрява структурата и четимостта на кода, но и позволява по-ефективно тестване и отстраняване на грешки, тъй като всяка част от архитектурата може да бъде изолирано тествана. Освен това, MVC подкрепя разработката на отговорни уеб приложения, които могат да се адаптират към различни устройства и размери на екрана, предоставяйки по-добро потребителско изживяване.

Изборът на MVC модела за моя дипломен проект е стратегическо решение, което отразява моето стремеж към създаването на добре организирано, мащабируемо и лесно за поддръжка уеб приложение. С този подход, аз се надявам да предоставя солидна основа за бъдещо разширение и оптимизация на приложението, като същевременно подчертавам значението на добрата софтуерна архитектура за успешната разработка на съвременни уеб решения.

Въведение

Моделът Model-View-Controller (MVC) е архитектурен шаблон, широко използван в разработката на софтуерни приложения. Този модел разделя приложението на три основни компонента - Модел, Изглед и Контролер, като по този начин осигурява организирана структура, която улеснява както разработката, така и поддръжката на сложни софтуерни системи. Целта на MVC е да раздели бизнес логиката от потребителския интерфейс, което води до по-ефективна разработка и възможност за многократно използване на код.

История на MVC

Концепцията за MVC беше въведена през 70-те години на XX век от Trygve Reenskaug, докато работеше в Xerox PARC. В този период, разработването на графични потребителски интерфейси (GUI) се нуждаеше от нов подход за структуриране на кода, който да позволи по-лесната му поддръжка и разширение. Така MVC се превръща в ключова архитектура, предлагаща решение на тези предизвикателства. С времето, MVC намира широко приложение в разработката на уеб приложения, като се адаптира и еволюира за да отговори на съвременните изисквания на софтуерната инженерия.

Основни компоненти на MVC

Модел

Моделът е сърцевината на архитектурния шаблон MVC, като представлява логическата структура на данните в софтуерното приложение. Той включва всичко, което се отнася до данните: способността да извлича, съхранява, обработва и валидира информация. В рамките на модела, данните могат да бъдат едновременно четени и писани, което позволява на приложението да извършва разнообразни операции, като например актуализиране на базата данни в отговор на потребителски действия. Моделът е строго отделен от потребителския интерфейс, което позволява на разработчиците да променят бизнес логиката без да засягат изгледа или контролера и обратно.

Изглед

Изгледът в MVC архитектурата е това, което потребителят вижда и взаимодейства с - потребителският интерфейс. Той включва всички елементи на интерфейса, като например текст, формуляри за въвеждане на данни, бутони и изображения. Изгледите в MVC са отговорни за представянето на модела в разбираем за потребителя формат. Важно е да се отбележи, че в MVC моделът може да има множество изгледи, свързани с един модел, позволявайки различни представяния на едни и същи данни.

Контролер

Контролерът действа като координатор между модела и изгледите. Той обработва входящи заявки от потребителите, взема решения въз основа на тези заявки (например, кое действие да бъде извършено), изисква информация от модела и избира подходящ изглед за представяне на тази информация. Контролерът играе ключова роля в управлението на потока на данни в приложението и е централен за реализацията на бизнес логиката.

Допълнителни аспекти на MVC архитектурата

Разделение на отговорности

Едно от най-големите предимства на MVC модела е четкото разделение на отговорности между различните компоненти. Това не само улеснява разработката и поддръжката на приложения, но и позволява на екипите да работят по-ефективно, като се специализират в определени аспекти на приложението - например, разработчиците могат да се фокусират върху бизнес логиката, докато дизайнерите работят върху интерфейса.

Многократна употреба на кода

При използването на MVC, кодът става по-модуларен и податлив на многократна употреба. Моделите могат да бъдат използвани повторно в различни части на приложението или дори в различни проекти. Това води до значително намаляване на времето за разработка и увеличава ефективността.

Гъвкавост при промени

Архитектурата на MVC позволява лесно да се правят промени в потребителския интерфейс без да се налага промяна на бизнес логиката и обратно. Това е особено полезно в динамичната среда на софтуерната разработка, където изискванията често се променят.

Предимства и недостатъци на MVC

Предимства

•Организация на кода: MVC спомага за чиста и организирана структура, което улеснява разбирането и поддръжката на кода

• Гъвкавост и многократна употреба: Разделянето на компонентите позволява лесното им повторно използване в различни части на приложението или дори в различни проекти.

•Улеснено тестване: MVC архитектурата улеснява модулното тестване на приложението, като позволява изолирано тестване на всяка компонента.

Недостатъци

•Комплексност: За малки проекти или начинаещи разработчици, архитектурата може да се стори прекалено сложна и трудна за осмисляне.

• Изпълнение: В някои случаи, допълнителната абстракция и разделение могат да доведат до забавяне на изпълнението на приложението.

Приложение на MVC в програмирането

MVC моделът намира широко приложение в съвременната софтуерна разработка, особено в уеб и мобилните приложения. Технологии като ASP.NET MVC, Spring MVC, Ruby on Rails и множество JavaScript фреймуърки като AngularJS и ReactJS, използват MVC шаблона като основа за своята архитектура, предлагайки разработчиците мощни инструменти за създаване на бързи, надеждни и лесно поддържаеми приложения.

**Заключение**

MVC моделът представлява фундаментален шаблон в софтуерната архитектура, който позволява разработването на ефективни, организирани и лесно поддържаеми приложения. С четкото разделение на отговорностите, подобряването на модуларността и гъвкавостта в разработката, MVC се утвърждава като предпочитан избор за множество разработчици по света. Въпреки предизвикателствата, свързани с неговата сложност и крива на обучение, предимствата, които предлага, го правят незаменим инструмент в арсенала на съвременната софтуерна инженерия.

Въведение в ASP.NET Core 3.1

ASP.NET Core 3.1 е продължение на усилията на Microsoft да предостави напреднал, отворен и крос-платформен фреймуърк за разработка на уеб приложения и API-та. Представен като еволюция на оригиналния ASP.NET, този фреймуърк е проектиран да отговори на съвременните изисквания за висока производителност, безопасност и гъвкавост в разработката на уеб и облачни базирани решения. Работещ еднакво добре на Windows, Linux и macOS, ASP.NET Core 3.1 улеснява разработчиците да създават и разпространяват приложения, които служат на широк спектър от устройства и платформи.

История и развитие на ASP.NET Core

ASP.NET Core е резултат от стремежа на Microsoft към иновации и подобрения в сферата на софтуерната инженерия. Стартиран като проект Kestrel, ASP.NET Core беше представен като пълна преработка на ASP.NET с цел постигане на по-висока производителност, модулност и крос-платформена съвместимост. С пускането на версия 3.1, Microsoft акцентира върху стабилността и дългосрочната поддръжка, предоставяйки на разработчиците надеждна основа за създаване на уеб приложения и услуги.

Ключови характеристики на ASP.NET Core 3.1

Поддръжка на крос-платформено разработване

С ASP.NET Core 3.1, разработчиците могат лесно да създават приложения, които работят на различни операционни системи. Това отваря вратата за по-широка аудитория и улеснява разработчиците да тестват и разпространяват своите приложения на разнообразни среди.

Използване на Razor Pages

Razor Pages предоставя по-прост и по-интуитивен синтаксис за създаване на динамични уеб страници с C#. Това улеснява разработчиците при създаването на уеб приложения, като едновременно с това поддържа структуриран и организиран код.

Blazor за интерактивни уеб интерфейси

Blazor е революционна технология в ASP.NET Core, която позволява използването на C# за създаване на интерактивни уеб интерфейси без необходимостта от JavaScript. Това дава възможност на .NET разработчиците да използват пълния потенциал на своите умения при създаването на богати клиентски приложения.

Подобрения в производителността

ASP.NET Core 3.1 внася значителни оптимизации в компилацията и изпълнението на кода, което води до по-бързо зареждане на приложенията и по-ефективно използване на ресурсите.

Акцент на сигурността

Фокусът върху сигурността е по-силен от всякога в ASP.NET Core 3.1, с усъвършенствани механизми за аутентификация, авторизация и защита от уеб заплахи, като XSS и CSRF атаки.

Предимства на ASP.NET Core 3.1

Гъвкавост и контрол

Модулната архитектура на ASP.NET Core 3.1 предоставя на разработчиците изключителен контрол върху своите приложения, като позволява лесна интеграция на сторонни библиотеки и услуги.

Съвместимост с контейнери

Оптимизацията на ASP.NET Core 3.1 за контейнери като Docker улеснява разработчиците в пакетирането, разпространението и мащабирането на своите приложения, като същевременно осигурява консистентност в различните среди за разработка и продукция.

Съвместимост

Оптимизацията на ASP.NET Core 3.1 за контейнери като Docker улеснява разработчиците в пакетирането, разпространението и мащабирането на своите приложения, като същевременно осигурява консистентност в различните среди за разработка и продукция.

Силно съобщество и поддръжка

Наличието на активно и поддържащо съобщество, заедно със силната поддръжка от страна на Microsoft, осигурява на разработчиците достъп до множество ресурси, водачи и най-добри практики за успешната разработка на приложения.

Примери за приложение на ASP.NET Core 3.1

ASP.NET Core 3.1 се използва за създаването на разнообразни уеб приложения, като:

•Динамични уеб сайтове и портали, предлагащи персонализирано съдържание за потребителите.

• RESTful API-та и микросервиси, които обслужват мобилни приложения, уебсайтове и други услуги.

•Интерактивни уеб приложения с Blazor, позволяващи създаването на богати и ангажиращи потребителски интерфейси.

•Сложни уеб базирани системи, които изискват висока производителност, скалируемост и интеграция с различни услуги и бази данни.

Потенциални недостатъци и предизвикателства

Въпреки множеството си предимства, преходът към ASP.NET Core 3.1 може да представлява предизвикателство за организации с големи наследени системи, изградени на по-стари версии на .NET Framework. Миграцията към ASP.NET Core изисква време, ресурси и потенциално преосмисляне на архитектурни решения, което може да бъде трудно за някои екипи.

Заключение

ASP.NET Core 3.1 предлага баланс между производителност, гъвкавост и сигурност, правейки го привлекателен избор за разработчици и организации, които искат да изградят съвременни уеб решения. С акцент върху крос-платформената съвместимост, поддръжката на контейнери и богатите функционалности за разработка на приложения, ASP.NET Core 3.1 остава ключов играч в света на уеб технологиите.