



**Природо-математическа гимназия
„Васил Друмев“
гр. Велико Търново**

ДИПЛОМНА РАБОТА

за придобиване на трета

степен на професионална

квалификация

**Тема: Изграждане на комплекс от методи, средства
и подходи при разработване на приложение
„ТОТО СИМУЛАТОР“**

Ученик:

Кристиян Красимиров Гешев

Ръководител консултант:

Милен Василев

Професия: „Приложен програмист“

Специалност: „Приложно програмиране“

Велико Търново, 2023 г.

СЪДЪРЖАНИЕ

1.УВОД.....	4
2.ТЕХНОЛОГИИ ЗА РЕАЛИЗИРАНЕ НА УЕБ-БАЗИРАНО ПРИЛОЖЕНИЕ	5
2.1 Особенности на уеб-базираните информационна система.....	5
2.2 Езици за реализиране на уеб приложения	7
2.2.1 HTML(HyperText Markup Language).....	7
2.2.2 CSS(Cascading Style Sheets)	10
2.2.3 C#.....	13
2.2.4 ASP.NET Core	15
2.2.5 Bootstrap.....	17
2.3 База данни и Система за управление на бази данни (СУБД)	18
2.4 Използвани средства за управление на базата данни	20
2.4.1 Релационен модел	20
2.4.2 SQL(Structured Query Language)	22
2.4.3 Microsoft SQL Server	23
2.4.4 Microsoft SQL Server Management Studio	24
2.4.5 Entity Framework Core	24
2.5 Архитектура (MVC)	25
3. РЕАЛИЗЦИЯ ПРИ РАЗРАБОТВАНЕ НА УЕБ-ПРИЛОЖЕНИЕ „ТОТО- СИМУЛАТОР“	27
3.1 Цели.....	27
3.2 Страници.....	27
3.2.1 Навигация	27
3.2.2 Начална страница.....	29
3.2.3 Списък с тиражи	29
3.2.*Методи за редакция и изтриване	32
3.2.4 Талони	34

3.2.5 Числа	35
3.2.6 Джакпоти.....	36
3.2.7 Администраторска страница (Потребители)	38
3.3 Модели и база данни	39
4. ЗАКЛЮЧЕНИЕ	40
Използвана литература	41

1.УВОД

„Тото симулатор“ 6 от 49 е забавна онлайн игра с некомерсиален характер. Задачата и е да даде възможност на потребителите да попълват електронен талон с избраните от тях числа, да симулира тегленето на печеливши числа, разпределение на печалби по групи, статистика, отразяване на печалби на потребител.

Симулаторът е базиран на лотарията, която е форма на игра, при която печалбите се обявяват и разпределят публично чрез жребий по предварително обявена схема, а получаването им зависи от познаването на определена цифра, комбинация от цифри, знак, фигура и други или от изтеглянето на печеливш предварително закупен фиш, талон или билет. Целта на участниците в лотарийните игри е паричната печалба. Най-желана е максималната печалба, наречена джакпот.

В „Тото симулатор“ потребителите попълват талон с 6 числа по техен избор. Числата са в диапазона от 1 до 49. Вече попълненият талон от потребителя се причислява към следващото теглене на тираж. При тегленето на тиража се избират 6 печеливши числа по случаен начин. След тегленето симулаторът определя спечелилите и разпределя печалбите. Печелившите талони са разпределени в четири групи спрямо колко числа съвпадат с тегленето. В I-ва група попадат участниците познали 6 числа; във II-ра – тези, които са познали 5 числа; в III-та – тези, които са познали 4 числа и в IV-та тези, които са познали 3 числа. Постъпленията за даден тираж се изчисляват на база единична цена на талон и броят на всички попълнени талони към този тираж. За фонд печалби се заделят $\frac{1}{2}$ от постъпленията в конкретния тираж. Определената печалба се разпределя по четирите групи като за I-ва група се заделя 40% от нея, докато за останалите групи се заделят по 20% от сумата за всяка. Ако няма победители в I-ва група, сумата предназначена за тях се прехвърля за следващият тираж под формата на джакпот. Джакпотът се занулява, ако има спечелили от I-ва група. Приема се обаче, че в останалите групи винаги има печеливши.

Симулаторът разполага и с администраторска страна. Потребителите, които имат ролята на „Администратор“ имат достъп до цялата база от данни. Те могат да създават, четат, редактират и изтриват данни от базата. Имат достъп и до страница с всички потребители, където могат да променят данни за всеки един, да му дадат и

премахват роли и също така да го изтрият от базата. Страницата разполага с начини за търсене, филтриране и странициране на списъка с потребители. Администраторът също задава интервала от време през, когото се генерира теглене на тираж. Всеки може да си направи регистрация в симулатора и след това да влиза в профила си чрез потребителско име и парола. Всеки нов профил е с роля на „Участник“. За да може един профил да получи ролята „Администратор“ трябва потребител, който вече е такъв да му даде ролята през администраторската страница със списъка на потребителите.

Участникът може да попълва талони и да ги разглежда в отделна страница. Той има право да преглежда само талони, които са попълнени от него самия. Изглед към всички талони имат само потребителите с роля „Администратор“. Участникът има и достъп до страници с всички тиражи, джакпоти и статистика за срещанията на дадени числа. Всяка една от тези страници има начини за филтриране, сортиране или търсене. Освен попълването на талони, участниците нямат правото да създават, променят или изтриват други данни от базата. Те също нямат достъп до преглеждането на информация за други потребители и техните талони.

2.ТЕХНОЛОГИИ ЗА РЕАЛИЗИРАНЕ НА УЕБ-БАЗИРАНО ПРИЛОЖЕНИЕ

2.1 Особенности на уеб-базираните информационна система

Уеб-базирани информационни системи, или системи, използващи интернет технологии, служат за предоставяне на информация и услуги на потребители или други информационни системи и приложения. Тези системи са създадени с цел да обработват и поддържат данни, като използват принципи, базирани на хипертекст. Такива информационни системи обикновено се състоят от едно или повече уеб приложения, специфични компоненти със задачи и информационни елементи, заедно с други не-уеб компоненти. Уеб-браузърът често служи като интерфейс за потребителя, докато базата данни функционира като основен източник на информация.

Благодарение на бързото развитие на уеб технологиите и растящия интерес

от потребители и разработчици, понятието за уеб сайт се превръща от просто набор от HTML страници в уеб-базирана информационна система (WIS). Такава система предоставя инструменти за достъп до сложни данни и интерактивни услуги чрез интернет. Електронни бизнес приложения, системи за управление на връзките с клиенти (CRM) и приложения за веригата за доставки са примери за WIS.

Макар да има сходства между традиционните и уеб-базирани информационни системи, има и съществени различия. Уеб-базирани системи създават специфични предизвикателства за разработчиците, които трябва да се справят с разнообразни мрежови условия, потребители, типове данни и други въпроси като управление на съдържание, представяне и удобство за потребителя. Въпреки че уеб-базирани информационни системи са платформено независими откъм предоставянето на информация, което допринася за тяхната популярност, уеб разработчиците трябва да вземат предвид множество мрежови особености при проектирането на системата. Анализът на потребителските изисквания за WIS представлява значително по-голямо предизвикателство за разработчиците, отколкото при традиционни информационни системи. Потребителите на традиционни системи често са служители, които работят в рамките на една организация или отдел. Напротив, анализирането на потребностите на потребителите на уеб-базирани информационни системи изисква разработчиците да разгледат много по-широк кръг от потребителски случаи и сценарии.

В заключение, уеб-базирани информационни системи са софтуерни системи, които използват интернет технологии за доставяне на информация и услуги до потребители и други информационни системи. Те се състоят от уеб приложения, информационни елементи и компоненти, които работят заедно, за да предоставят достъп до сложни данни и интерактивни услуги. Разработчиците на такива системи трябва да се справят с предизвикателства, свързани с различни мрежови условия, потребители и типове данни, като същевременно осигурят управление на съдържание, представяне и удобство за потребителя.

2.2 Езици за реализиране на уеб приложения

2.2.1 HTML(HyperText Markup Language)

HTML е съкращение от "HyperText Markup Language" и е стандартен език за маркиране на уеб страници. Той се използва за създаване на съдържание и структура на уеб страници, като определя текста, изображенията, линковете, списъците, формите и други елементи на страницата.

Елементите на страницата се маркират чрез HTML тагове. Таговете са написани между скоби < > и обикновено имат отварящ и затварящ таг. Например, за да дефинираме заглавие на страницата, използваме тага <head> и <title>. Пример:

```
<head>
  <title>Заглавие на страница</title>
</head>
```

Фигура 1. Пример за HTML таг

Таговете в HTML могат да бъдат с различни атрибути, които предоставят допълнителна информация за елемента. Атрибутите се записват в началния таг и могат да се използват за различни цели, като настройка на цветове, определяне на размери или посочване на линкове към други страници.

```

```

Фигура 2. Пример за атрибут в HTML

В този пример, src е атрибут, който определя пътя към изображението, а alt атрибутът предоставя текстово описание на изображението за потребителите, които не могат да видят изображението.

HTML предоставя много елементи, които могат да бъдат използвани за създаване на различни компоненти на страницата. Елементите за текст са много, включително заглавия, параграфи, списъци, таблица и т.н. Има и специализирани елементи като форми, които могат да съдържат текстови полета, радио бутони, падащи менюта и т.н. Тези елементи могат да се използват за създаване на форми за вход на потребители, като контактна форма или форма за регистрация.

HTML също така предоставя много възможности за създаване на линкове между различни страници и места в рамките на същата страница. Линковете се създават чрез елемента `<a>` и могат да се използват за да свържат различни страници или места в рамките на същата страница. Линковете могат да имат текстово съдържание или изображения и се определят чрез атрибута `href`, който указва адреса на страницата или местоположението в страницата, към която да се отиде при кликуване върху линка.

```
<a href="http://www.example.com">Линк към сайта на Example</a>
```

Фигура 3. Пример за линк в HTML

HTML е език, който се използва за определяне на структурата на уеб страници и е важна част от уеб разработката. Той е динамичен език, който може да бъде използван в комбинация с други технологии като CSS за оформление на страници и JavaScript за добавяне на интерактивност и динамично поведение на уеб сайтовете.

Семантичност в HTML

Семантичността в HTML се отнася до използването на елементи, които имат ясно определено значение или цел, което подпомага както разработчиците, така и браузърите да разберат структурата и съдържанието на уеб страница. Семантичните елементи в HTML5 дават по-ясна представа за различните части на уеб страницата, като навигация, заглавие, основно съдържание, странично съдържание и др.

Използването на семантични елементи предлага следните предимства:

- Четимост: Семантичните тагове улесняват разработчиците да разберат структурата на уеб страницата, като правят кода по-чист, организиран и лесен за разбиране.
- Подобрена SEO (оптимизация за търсачки): Търсачките като Google използват семантичните елементи за да разберат структурата и

съдържанието на уеб страницата, което може да подпомогне класирането на сайта в резултатите от търсенето.

- **Достъпност:** Семантичните елементи помагат на технологиите за подпомагане, като четци на екрана, да разберат и интерпретират съдържанието на уеб страницата, което прави сайта по-достъпен за хора с увреждания.

Някои от основните семантични елементи в HTML5 включват:

- **<header>:** Определя заглавната част на уеб страница или секция.
- **<nav>:** Определя навигационната част на уеб страница, която съдържа линкове за навигация.
- **<main>:** Определя основното съдържание на уеб страницата, което е уникално за конкретната страница.
- **<article>:** Определя самостоятелен, независим контент, който може да бъде разбран от контекста на страницата, като например новина или блог пост.
- **<section>:** Определя раздел в рамките на уеб страницата, който може да съдържа заглавие и други HTML елементи.
- **<aside>:** Определя страничното съдържание, което е свързано с основното съдържание на страницата, но не е непосредствено част от него. Това може да включва бележки, реклами или допълнителни ресурси.
- **<footer>:** Определя долната част на уеб страница или секция, която може да съдържа информация като автор, копирайт или линкове към други страници.
- **<figure> и <figcaption>:** Определят графичен или мултимедиен елемент (например изображение, диаграма или видео) и свързаното с него описание или легенда.
- **<time>:** Определя време или дата и може да бъде използван за форматиране или стилизиране на дати и часове в различни формати.

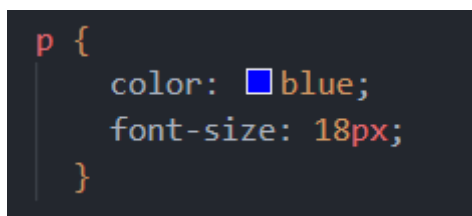
Използването на семантични елементи в HTML осигурява по-добра структура на кода, подобрява четимостта, оптимизира SEO и подпомага достъпността. Тези елементи предоставят ясна идентификация на различните части на уеб страницата,

което позволява на разработчиците, браузърите и технологиите за подпомагане да работят по-ефективно и да осигурят по-добро потребителско изживяване.

2.2.2 CSS(Cascading Style Sheets)

CSS е език за оформление на уеб страници. Той се използва за определяне на външния вид и стил на HTML елементите. CSS позволява да се използва цвят, шрифтове, размери, позиции, изображения и други елементи, за да се стилизират уеб страници.

Синтаксисът на CSS се състои от правила, които определят как трябва да се изглеждат определени HTML елементи. Правилата са съставени от две основни части - селектор и декларации. Селекторът определя кой HTML елемент или група от елементи трябва да бъдат стилизирани, а декларациите определят как трябва да изглеждат тези елементи.



```
p {  
    color: blue;  
    font-size: 18px;  
}
```

Фигура 4. Пример за правило в CSS

В този пример `p` е селекторът, който избира всички HTML елементи на страницата, които са параграфи. Декларациите, които следват селектора, определят, че текстът на параграфите трябва да бъде в син цвят и с размер на шрифта 18 пиксела.

Основните концепции в CSS включват:

1. Селектори - като елементарни, класове, идентификатори, комбинирани и псевдо-класове. Те позволяват да се изберат определени елементи на страницата, към които да се приложат стиловете.
2. Свойства - като цвят, шрифтове, размери, позиции и други, които дефинират конкретни характеристики за определен HTML елемент.

3. Каскадност - принципът, по който CSS декларации се наследяват и препокриват в зависимост от тяхното местоположение и приоритет във stylesheet-a.
4. Модуларност - възможността да се разделят стиловете на страницата на отделни файлове и да се прилагат при нужда.
5. Бокс модел - начина, по който HTML елементите се представят като прости кутии със зададена ширина, височина, padding, border и margin.
6. Респонзивен дизайн

Респонзивен дизайн е техника в CSS, която позволява на уеб страници да се приспособят към различни размери на екрана на устройства, като мобилни телефони, таблети, лаптопи и десктоп компютри. Това се постига чрез използване на медийни заявки (media queries), които проверяват размера на екрана на устройството и при нужда прилагат различни CSS стилове за определени елементи на страницата.

```
@media only screen and (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

Фигура 5. Пример за медийна заявка

Тази медийна заявка проверява дали широчината на екрана на устройството е по-малка от 600 пиксела и ако е така, променя фоновия цвят на страницата на светло синьо.

Освен това, в CSS има и други концепции като позициониране, флексбокс, гридове и трансформации, които позволяват създаването на сложни дизайни и анимации на уеб страниците.

В крайна сметка, CSS е важен инструмент в уеб разработката, който позволява на дизайнерите и разработчиците да създават красиви и функционални

уеб страници, които се приспособяват към нуждите на потребителите и устройствата, на които се достъпват.

Използването на CSS (Cascading Style Sheets) предлага редица предимства в сравнение с инлайн стилове или оформление на уеб страница само с HTML. Ето някои от основните предимства на CSS:

1. Отделение на съдържание и оформление: CSS позволява разделяне на съдържанието на уеб страницата (написано на HTML) от оформлението (стиловете на CSS). Това улеснява поддръжката и разработката, като позволява промени в дизайна без да засяга HTML структурата.
2. Централизиране и последователност на стиловете: CSS стиловете могат да бъдат съхранени в отделен файл и да се прилагат към много HTML страници. Това улеснява актуализирането и поддръжката на стиловете, осигурявайки последователно оформление на всички страници на уеб сайта.
3. По-малко код и по-бързо зареждане на страницата: CSS използва компактен синтаксис и може да се прилага към множество елементи с едно правило. Това намалява обема на кода, което води до по-бързо зареждане на страниците.
4. Повторно използване на стилове: С помощта на CSS, разработчиците могат да създадат библиотеки от стилове, които могат да бъдат лесно преизползвани в различни проекти, което спестява време и усилия при разработка на нови уеб страници.
5. Адаптивен и реактивен дизайн: CSS предлага инструменти за създаване на адаптивни и реактивни дизайни, които се адаптират към различни размери на екрана, устройства и ориентации, като медийни заявки и гъвкави мрежови модели.
6. Анимации и транзиции: CSS3 включва функции за анимации и транзиции, които позволяват на разработчиците да създават сложни и привлекателни анимации и ефекти, без нужда от допълнителни скриптове или библиотеки.
7. Улеснена достъпност: CSS позволява на разработчиците да създават уеб страници, които са по-достъпни за хора с увреждания или ограничения.

Стиловете могат да бъдат адаптирани според нуждите на потребителите, като например увеличение на размера на шрифта или контраста за хора със зрителни проблеми.

8. Контрол на стиловете за печат: CSS позволява да се определят специфични стилове за печат на уеб страници, така че потребителите могат да разпечатат информацията в подходящ и удобен формат.
9. Поддръжка на различни уеб браузъри: CSS предоставя механизми за използване на различни стилове, в зависимост от поддръжката на различните уеб браузъри. Това позволява на разработчиците да осигурят стабилно и приятно преживяване за потребителите, независимо от използвания браузър или версия.
10. Лесен за научаване и разбиране: CSS е език с относително прост синтаксис и логика, което го прави достъпен и лесен за научаване за начинаещи разработчици. Това означава, че можете бързо да започнете да работите с CSS и да създавате персонализирани уеб дизайни.

CSS може да бъде вграден директно в HTML файла чрез `<style>` таг, включен като външен файл с `<link>` таг или приложен върху отделен елемент чрез атрибута `style`. Разделението на стиловете от HTML структурата допринася за подобро разделение на отговорностите, по-добра четимост и поддръжка на кода, и по-лесно преизползване на стилове.

2.2.3 C#

C# (произнася се "C-sharp") е модерен, обектно-ориентиран програмен език, разработен от Microsoft. Той е част от платформата .NET и е създаден от Андерс Хейлсберг през 2000 година. C# е силно вдъхновен от езика C++ и Java, като целта му е да предложи лесен за използване и изразителен език за разработка на различни видове приложения, включително уеб, мобилни, настолни и облачни решения.

Основни характеристики на C# включват:

- Обектно-ориентирано програмиране: C# е базиран на принципите на обектно-ориентираното програмиране (ООП), което включва енкапсулация, наследяване и полиморфизъм.

- Силна типизация: C# изисква ясно определени типове на данни, което означава, че разработчиците трябва да определят типа на всеки обект или променлива. Това подпомага намаляване на грешките и улеснява откриването на проблеми по време на компилация.
- Управление на паметта: C# използва автоматично управление на паметта чрез "събиране на боклука" (garbage collection), което автоматично освобождава памет, когато обектите не са вече нужни. Това намалява риска от утечки на памет и улеснява управлението на ресурсите.
- Изключения: C# предоставя механизми за обработка на грешки чрез изключения, което улеснява управлението на грешки и предоставя по-добра информация за възникнали проблеми.
- Ламбда изрази и LINQ: C# включва поддръжка за ламбда изрази и Language Integrated Query (LINQ), което позволява на разработчиците да използват декларативно програмиране и работа с данни от различни източници по един консистентен и ефективен начин.
- Асинхронно програмиране: C# предлага поддръжка за асинхронно програмиране, което позволява на приложенията да изпълняват задачи паралелно или да изпълняват несвързани операции, без да блокират главната програма. Това улеснява създаването на гладко работещи и отзивчиви приложения, особено при работа с външни ресурси или операции, които отнемат много време.
- Атрибути и метаданни: C# позволява на разработчиците да използват атрибути, които са маркери за метаданни, приложени към елементи на кода, като класове, методи или свойства. Тези атрибути могат да бъдат използвани за предоставяне на допълнителна информация или инструкции към компилатора или средата на изпълнение.
- Интероперабилност с други езици и платформи: C# може да си взаимодейства с код, написан на други програмни езици, и да използва съществуващи библиотеки и компоненти. Това улеснява интегрирането на C# в съществуващи проекти и позволява разработчиците да използват широка гама от инструменти и технологии.

- Cross-platform разработка: Чрез .NET Core и .NET 5 (и по-нови версии), C# може да се използва за създаване на приложения, които работят на различни платформи, включително Windows, macOS и Linux.
- Богата стандартна библиотека: C# има богата стандартна библиотека, която предоставя редица функции и класове, които позволяват на разработчиците да създават мощни и функционални приложения. Тази библиотека включва поддръжка за работа с файлове, мрежи, дати и времена, колекции, математика, криптография и много други.

C# се използва широко в индустрията и е особено популярен в областта на разработването на видеоигри със средата Unity. C# продължава да се развива и подобрява, като се адаптира към нови технологии и практики в разработката на софтуер.

2.2.4 ASP.NET Core

ASP.NET Core е отворен код, крос-платформен и модерен уеб-разработка framework, разработен от Microsoft. Той е изграден върху .NET Core, който предоставя обща база код за разработване на високопроизводителни и мащабируеми приложения за различни платформи, включително Windows, Linux и macOS. ASP.NET Core е наследник на класическия ASP.NET и е проектиран да предостави по-добра производителност, гъвкавост и сигурност.

Някои от ключовите характеристики на ASP.NET Core включват:

- Крос-платформеност: ASP.NET Core може да се използва за разработване на приложения, които работят на различни операционни системи, включително Windows, Linux и macOS. Това означава, че разработчиците могат да използват един и същ код за разработване на приложения за различни платформи.
- Модулна архитектура: ASP.NET Core използва модулна архитектура, която позволява разработчиците да добавят или премахват функционалност чрез добавяне или премахване на NuGet пакети. Това позволява по-гъвкаво и леко разработване на приложения.

- Висока производителност: ASP.NET Core е проектиран да предоставя по-добра производителност в сравнение с предишни версии на ASP.NET. Той използва Kestrel, много бърз и лек HTTP сървър, базиран на libuv, което подобрява производителността на приложенията.
- Контейнеризация и микросервиси: ASP.NET Core поддържа разработване на микросервиси и контейнеризация с Docker, което улеснява разработването, тестването и разпространението на приложенията.
- Сигурност: ASP.NET Core предоставя разширени възможности за сигурност, включително аутентикация, авторизация и защита на данни. Той също така поддържа GDPR и други международни стандарти за защита на личните данни.
- Razor Pages: Razor Pages е нова функционалност в ASP.NET Core, която улеснява разработването на уеб приложения чрез по-проста и организирана структура на кода. Razor Pages са базирани на Razor синтаксиса и позволяват създаването на динамични уеб страници, които съчетават HTML, CSS и C# код.
- Dependency Injection: ASP.NET Core има вградена поддръжка за Dependency Injection (DI), което е популярен модел за разработка на софтуер, който подпомага разделението на отговорностите. DI улеснява управлението на зависимости, тестването и мащабирането на приложенията.
- Среди за разработка: ASP.NET Core може да се разработва и тества с помощта на различни среди за разработка (IDE) като Visual Studio, Visual Studio Code и Visual Studio for Mac, което позволява на разработчиците да изберат най-подходящата за тях среда за работа.
- Middleware компоненти: В ASP.NET Core, middleware компонентите са използвани за обработка на HTTP заявки и отговори. Те могат да се конфигурират и добавят към приложението, като създават възможност за модификация на входящите и изходящите данни.
- SignalR: SignalR е библиотека за ASP.NET Core, която позволява реално време комуникация между клиент и сървър чрез WebSockets и други транспортни механизми. Това улеснява създаването на интерактивни и

динамични уеб приложения, като игри, чат приложения и други.

Накратко, ASP.NET Core е мощен и гъвкав уеб-разработка framework, който предоставя множество функционалности и инструменти за създаване на съвременни, високопроизводителни и мащабируеми уеб приложения. Отвореният код и крос-платформената поддръжка правят ASP.NET Core привлекателен избор за разработчици, които искат да създават висококачествени уеб приложения и услуги, работещи на различни операционни системи.

2.2.5 Bootstrap

Bootstrap е популярна библиотека за разработка на интерфейси, която се използва за създаване на съвременни и адаптивни веб-страници и приложения. Тя е създадена от Twitter и е с отворен код, което означава, че може да бъде свободно използвана, променяна и разпространявана от разработчиците. Bootstrap е базиран на HTML, CSS и JavaScript и предоставя готови стилове, компоненти и скриптове, които позволяват на разработчиците бързо да изграждат визуално атрактивни и функционални веб-страници.

Едно от ключовите предимства на Bootstrap е, че той е мобилно първо, което означава, че е създаден с основна цел да функционира оптимално на различни мобилни устройства. Това се постига чрез използването на адаптивни решетки, които автоматично променят размера и оформлението на елементите в зависимост от размера на екрана на устройството. Това прави разработката на мобилни приложения по-лесна и по-бърза, тъй като разработчиците не трябва да създават отделни версии на сайта за различните устройства и размери на екрана.

Освен адаптивните решетки, Bootstrap включва множество готови компоненти, като навигационни ленти, форми, бутони, модални прозорци и карусели, които могат лесно да се добавят към веб-страницата чрез HTML и CSS класове. Това прави процеса на дизайн и стилизиране на веб-страницата бърз и ефективен, като позволява на разработчиците да се съсредоточат върху функционалността на приложението.

Скриптовите на JavaScript в Bootstrap осигуряват допълнителни интерактивни функции, като отваряне и затваряне на модални прозорци, анимации и превключватели, които подобряват потребителското изживяване. Тези скриптове са

основани на jQuery, популярна JavaScript библиотека, която улеснява манипулацията на DOM елементите и обработката на събития във веб приложенията.

Bootstrap също така предлага разширяемост и гъвкавост. Разработчиците могат да модифицират и персонализират стиловете и компонентите, за да отговарят на индивидуалните нужди и предпочитания на проекта, като използват предварително дефинирани променливи или създават свои собствени CSS правила. Това означава, че Bootstrap може да бъде използван за създаване на уникални и персонализирани дизайни, без да жертвате предимствата на удобството и бързината на разработката.

Един от недостатъците на Bootstrap може да бъде, че някои сайтове и приложения, които използват библиотеката, могат да изглеждат подобни поради използването на същите готови компоненти и стилове. Въпреки това, с достатъчно креативност и персонализация, разработчиците могат да създадат уникални и интересни дизайни, които се отличават от стандартните Bootstrap теми.

Общо взето, Bootstrap е мощен инструмент за разработка на веб интерфейси, който предлага гъвкавост, бързина и удобство на разработчиците. С неговата помощ, може лесно да се създават адаптивни и функционални веб-страници и приложения, които отговарят на съвременните стандарти за дизайн и потребителско изживяване.

2.3 База данни и Система за управление на бази данни (СУБД)

База данни е структурирано съхранение на информация, което позволява лесен достъп, модифициране и управление на данните. Тя се състои от съвкупност от данни, които са организирани във формат, удобен за обработка и анализ от компютърни програми. Базите данни са съществен компонент на информационните системи и приложения, тъй като те предоставят централизирано място за съхранение на информация, която може да бъде използвана от различни приложения и потребители.

Структурите в базите данни са организационни схеми, които определят начина, по който данните се съхраняват и манипулират в базата данни. Те включват следните основни компоненти:

- Таблици (или колекции) - това са основните структурни единици в базите

данни, които съхраняват данни в редове и колони (в реляционните бази данни) или в документи (в NoSQL бази данни).

- Колони (или атрибути) - те представляват различните категории информация, които се съхраняват за всеки запис в таблицата. Колоните имат тип данни, който определя вида на информацията, която може да се съхранява в тях.
- Редове (или записи) - те представляват индивидуални обекти или събития, за които се съхранява информация в базата данни. Всяко поле в реда съдържа стойност за съответната колона.
- Ключове - това са специални колони, които се използват за идентификация на записите и осигуряване на връзки между таблиците. Примери за ключове включват примарни ключове, които са уникални за всяка таблица, и външни ключове, които свързват таблиците в реляционната база данни.
- Индекси - те са структури, които оптимизират скоростта на търсенето и извличането на данни от базата данни. Индексите могат да се създадат върху една или повече колони в таблицата и подобряват производителността на заявките.
- Ограничения - това са правила, които определят валидните стойности и взаимоотношения между данните в базата данни. Ограниченията могат да включват уникалност, непразни стойности, допустими интервали или специфични формати.

Система за управление на бази данни (СУБД) е софтуер, който служи за управление и контрол на базите данни. СУБД осигурява средства за създаване, модифициране, изтриване и извличане на данни от базата, като предоставя различни функции, като сигурност, резервно копие, възстановяване и оптимизация на производителността. Системите за управление на бази данни също така осигуряват интерфейс между приложенията и базите данни, като позволяват на програмистите да работят с данните по един структуриран и абстрактен начин.

Съществуват различни видове СУБД, които отговарят на различните нужди и изисквания на информационните системи. Някои от основните категории включват:

- Релационни СУБД (RDBMS) - тези системи използват релационна модел на

данните и се основават на таблична структура, където данните се съхраняват в редове и колони. Примери за реляционни СУБД са MySQL, PostgreSQL, Microsoft SQL Server и Oracle Database.

- NoSQL СУБД - това са системи, които не използват реляционен модел и са създадени за обработка на големи количества разпределени данни. Те обикновено се характеризират с гъвкавост, хоризонтално скалиране и висока производителност. Примери за NoSQL СУБД включват MongoDB, Cassandra, Couchbase и Redis.
- Обектно-ориентирани СУБД - тези системи използват обектно-ориентиран модел за представяне на данните, като съхраняват и манипулират обекти, които съответстват на реални същности и взаимодействия в приложението. Обектно-ориентирани СУБД са по-подходящи за приложения, които изискват сложни взаимодействия между обектите, като например CAD системи, симулации и т.н. Примери за обектно-ориентирани СУБД включват db4o, ObjectDB и Versant Object Database.
- NewSQL СУБД - това са съвременни системи, които се стремят да комбинират най-добрите характеристики на реляционните и NoSQL бази данни. Те предлагат скалируемост и гъвкавост, подобни на NoSQL системите, но същевременно поддържат традиционни транзакционни и консистентностни свойства на реляционните СУБД. Примери за NewSQL СУБД са CockroachDB, Google Spanner и NuoDB.

Изборът на подходяща система за управление на бази данни зависи от изискванията на приложението, характеристиките на данните и предпочитанията на разработчиците. Важно е да се направи добър анализ на нуждите на проекта, за да се определи най-добрият тип СУБД и да се гарантира ефективност, сигурност и надеждност на информационната система.

2.4 Използвани средства за управление на базата данни

2.4.1 Реляционен модел

Реляционните бази данни са системи за съхранение и управление на информация, които използват реляционния модел, предложен от Едгар Франк Код в

1970 г. Релационният модел е изграден върху концепцията за таблица, състояща се от редове и колони, като всяка колона представлява атрибут (характеристика), а всяко съчетание на атрибутите образува ред, наречен запис.

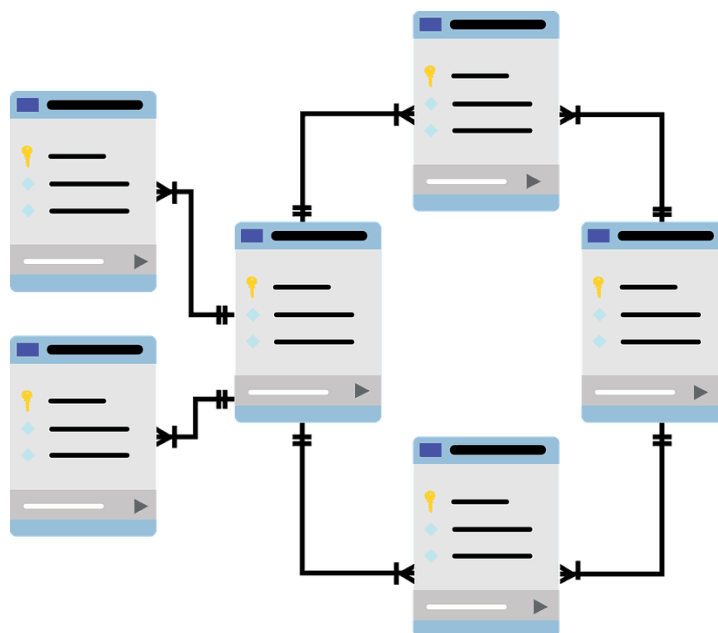
В релационните бази данни данните се организират в таблици, които се наричат релации. Табlici в релационните бази данни могат да се свързват помежду си чрез ключове (първични и външни). Първичният ключ е уникален идентификатор за всеки запис в таблицата, докато външният ключ е колона или множество от колони, които отговарят на първичния ключ на друга таблица. Свързването на таблици чрез ключове позволява извличането и модифицирането на информацията по ефективен и структуриран начин.

Основните предимства на релационните бази данни включват:

- Структурираност: Релационните бази данни предоставят ясна и логична структура на данните, което улеснява разбирането, обработката и анализа на информацията.
- Гъвкавост: Релационните бази данни позволяват лесно добавяне, промяна и изтриване на таблици и атрибути, без да се налага промяна на цялата структура на базата данни.
- Цялостност на данните: Релационните бази данни осигуряват механизми за поддържане на цялостта на данните, като гарантират уникалност на първичните ключове и коректност на взаимоотношенията между таблици.
- Ефективно управление на данните: Релационните бази данни поддържат SQL, стандартен език за управление на данните, който позволява изпълнение на сложни заявки и анализ на информацията.
- Сигурност: Релационните бази данни предлагат различни нива на сигурност, като управление на достъпа до данните и функционалностите на базата данни, както и криптиране на информацията. Това позволява да се контролира достъпът до чувствителни данни и да се предотвратят злоупотреби.
- Масшабируемост: Релационните бази данни могат да се мащабират хоризонтално и вертикално, за да се справят с растящите изисквания за

обработка и съхранение на информация. Те могат да се разполагат на различни хардуерни платформи и да се използват в облачни или хибридни инфраструктури.

- **Оптимизация:** Релационните бази данни разполагат с механизми за оптимизация на производителността, като индекси, кешове и оптимизатори на заявките, които улесняват бързото и ефективно изпълнение на заявките.
- **Транзакционност:** Релационните бази данни поддържат транзакционна обработка на данните, което гарантира цялостта и консистентността на информацията при съвместно изпълнение на множество операции.
- **Възстановяване и архивиране:** Релационните бази данни предоставят възможности за възстановяване и архивиране на данните в случай на срив на системата или загуба на информация. Това осигурява защита на данните и способност за продължаване на работата след хардуерни или софтуерни проблеми.



Фигура 6. Графика на релационна схема, представяща връзките между полетата в отделните модели

2.4.2 SQL(Structured Query Language)

SQL (Structured Query Language) е стандартизиран програмен език, използван за управление на релационни бази данни и извършване на различни операции върху

данните, съхранени в тях. SQL е специализиран език, разработен за опростяване на задачите, свързани с добавяне, промяна, изтриване и извличане на данни от бази данни.

Основните компоненти на SQL са:

- DDL (Data Definition Language) - операции за дефиниране и модифициране на структурата на базата данни, включително създаване, промяна и изтриване на таблици и схеми. Примерни DDL команди са: CREATE, ALTER и DROP.
- DML (Data Manipulation Language) - операции за работа с данните, съхранени в базата данни, включително добавяне, промяна, изтриване и извличане на информация. Примерни DML команди са: SELECT, INSERT, UPDATE и DELETE.
- DCL (Data Control Language) - операции, свързани с управлението на правата за достъп до данните в базата данни, включително наделяване на привилегии и контрол на достъпа. Примерни DCL команди са: GRANT и REVOKE.
- TCL (Transaction Control Language) - операции, свързани с управлението на транзакции и осигуряването на консистентност и изолация на данните. Примерни TCL команди са: COMMIT, ROLLBACK и SAVEPOINT.
- SQL е широко приет стандарт, поддържан от множество системи за управление на бази данни (СУБД) като Oracle, Microsoft SQL Server, PostgreSQL, MySQL и други. Въпреки че основният синтаксис на SQL е сходен между различните СУБД, могат да има специфични разширения и различия в детайлите.

2.4.3 Microsoft SQL Server

Microsoft SQL Server е релационна система за управление на бази данни (РСУБД), разработена от Microsoft. Тя предоставя скалируемо, безопасно и надеждно съхранение на данни, както и мощни инструменти за управление и анализ на тези данни. SQL Server използва стандартния език за заявки към бази данни SQL за изпълнение на операции с данни и администриране на базата данни.

SQL Server е широко използван в разнообразни бизнес приложения, като уеб сайтове, корпоративни информационни системи, аналитични платформи и много

други. Той се интегрира лесно с други продукти на Microsoft, като Visual Studio и Azure, което го прави изключително популярен избор сред разработчиците и администраторите на базиданни.

2.4.4 Microsoft SQL Server Management Studio

Microsoft SQL Server Management Studio (SSMS) е интегрирана среда за управление, разработка и администриране на релационни бази данни и SQL Server инстанции. Този мощен инструмент предоставя графичен интерфейс за управление на бази данни, изпълнение на SQL заявки, настройка на сигурността, мониторинг на производителността и много други задачи, свързани с управлението на SQL Server.

SSMS е изключително полезен за администраторите на бази данни, тъй като те могат да управляват лесно и ефективно както единични, така и множество SQL Server инстанции. Разработчиците могат да използват SSMS за създаване, тестване и отстраняване на грешки в SQL скриптове и за анализ на структурата на базата данни.

SSMS поддържа разширения и плъгини, които позволяват на потребителите да персонализират и разширят функционалността на средата според своите нужди.

2.4.5 Entity Framework Core

Entity Framework Core (EF Core) е лека, простираща се и кросплатформена версия на популярния Object-Relational Mapping (ORM) фреймуърк Entity Framework от Microsoft. EF Core позволява на разработчиците да работят с бази данни, използвайки .NET обекти, без да е необходимо да пишат код за директно взаимодействие с базата данни.

EF Core поддържа множество бази данни, включително релационни и NoSQL бази данни. С EF Core разработчиците могат да използват различни подходи като Code First, Database First и Model First, които определят как се създава моделът на базата данни и как се поддържа в хода на разработката.

Основните характеристики на EF Core включват:

- LINQ (Language Integrated Query) поддръжка: Това позволява на разработчиците да използват езика C# за създаване и изпълнение на заявки към базата данни.

- **Change Tracking:** EF Core проследява промените в обектите и ги синхронизира с базата данни при извикване на `SaveChanges()` метода.
- **Миграции:** EF Core поддържа миграции, които позволяват на разработчиците да автоматизират процеса на актуализация на схемата на базата данни, когато моделът се промени.
- **Поддръжка на кросплатформени приложения:** EF Core може да се използва на различни платформи, включително Windows, Linux и macOS, както и във всички версии на .NET Core и .NET 5+.

Въпреки че EF Core има много общи черти с класическия Entity Framework, той е по-бърз, по-модулен и по-гъвкав, което го прави подходящ за съвременни разработки на уеб и кросплатформени приложения.

2.5 Архитектура (MVC)

MVC (Model-View-Controller) архитектурата е дизайн патерн за софтуерно разработване, който разделя кода на три основни компонента: модел, изглед (view) и контролер. Целта на MVC е да организира кода по начин, който подобрява мащабируемостта, гъвкавостта и поддръжката на приложенията.

Модел (Model): Моделът представлява данните и логиката за работа с тези данни в приложението. Той се занимава с извличане, съхранение и обработка на информацията, както и със задаване на правилата и ограниченията, свързани с данните. Моделът е отговорен за комуникацията с базите данни, външни API-та и други източници на информация.

Изглед (View): Изгледът е частта от приложението, която се занимава с представянето на данните на потребителя. Той визуализира информацията от модела и предоставя интерфейс за взаимодействие с потребителя. Изгледът може да включва HTML, CSS, JavaScript и други технологии за уеб разработка, които се използват за структуриране и стилизиране на потребителския интерфейс.

Контролер (Controller): Контролерът управлява взаимодействието между модела и изгледа. Той приема входни данни от потребителя, получени чрез изгледа, и ги обработва, използвайки логиката, дефинирана в модела. След това контролерът актуализира изгледа, за да отрази промените в данните. Контролерът служи като

свързваща връзка между модела и изгледа и координира тяхното взаимодействие.

MVC архитектурата е широко използвана в уеб разработката и е основа на много популярни рамки за програмиране, като например Ruby on Rails, Django, ASP.NET MVC и други. Използването на MVC подпомага разработчиците да създават по-чист код, като разделят функционалността на отделни компоненти, което допринася за лесното поддържане и разширяване на приложенията.

Освен предимствата, които споменахме, MVC архитектурата има и други ползи:

- Разделението на отговорности: Тъй като моделът, изгледът и контролерът имат различни функции, това позволява на разработчиците да се фокусират върху един аспект на приложението на всяка една стъпка от процеса на разработка.
- Лесна тестваемост: Всяка част от MVC архитектурата може да се тества независимо, което подобрява качеството на кода и намалява вероятността за грешки.
- Гъвкавост при промяна на дизайна или функционалността: Разделението на компонентите в MVC архитектурата означава, че промените в дизайна или функционалността на приложението могат да се извършват по-лесно, без да се налага промяна на целия код.
- Улеснена работа в екип: Разделението на кода на три отделни компонента позволява на разработчиците да работят по различни части на приложението едновременно, без да се притесняват от конфликти в кода.

Въпреки предимствата на MVC архитектурата, е важно да отбележим, че тя може да не е подходяща за всички видове проекти. За малки приложения или приложения с прости потребителски интерфейси, използването на MVC може да доведе до излишно усложняване на кода. В такива случаи, разработчиците могат да предпочетат по-леки и по-прости архитектурни модели.

В заключение, MVC архитектурата е мощен и гъвкав дизайн патерн, който помага на разработчиците да създават устойчиви и лесно поддържани приложения. Този подход е особено подходящ за създаване на сложни уеб приложения и се използва от множество популярни програмни рамки.

3. РЕАЛИЗЦИЯ ПРИ РАЗРАБОТВАНЕ НА УЕБ-ПРИЛОЖЕНИЕ „ТОТО-СИМУЛАТОР“

3.1 Цели

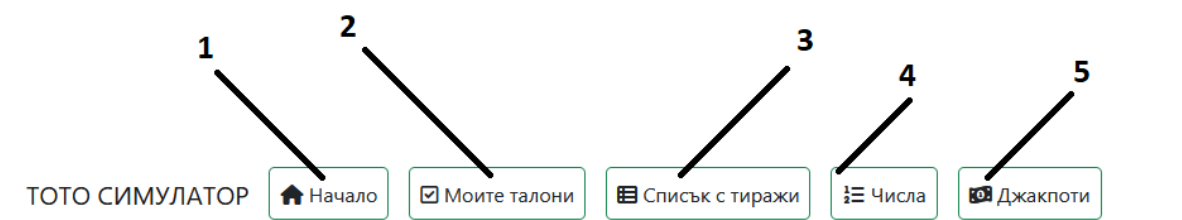
1. Забавление и ангажираност на потребителите: Основната цел на „ТОТО-СИМУАЛТОР“ уебсайта е да предостави забавление и интерактивно преживяване на потребителите. Симулаторът предоставя възможност за попълване талони, преглед на информация за тиражите, талоните, джакпотите и числата.
2. Интуитивен и лесен за използване интерфейс: Уебсайтът предлага интуитивен и лесен за използване интерфейс, който позволява на потребителите бързо да се адаптират и започнат да използват симулатора. Администраторите също имат удобен интерфейс, позволявайки им да модифицират данните в базата.
3. Информация и статистика: Приложението предлага информация за всички компоненти от тотото като талони, тиражи, джакпоти и числа. Потребителите могат да разглеждат отминали тиражи и наградните им фондове, джакпотите и техния растеж, числата и броя срещания на всяко едно.
4. Възможност за контрол върху данните: Потребителите с роля „Администратор“ имат правото да преглеждат, създават, променят и изтриват данни от базата. Те също управляват информацията за останалите потребители и техните роли.
5. Сигурност и поверителност: Потребителската информация и данни се съхраняват безопасно в базата данни и паролите задължително се хешират.
6. Подредба на изгледа: Потребителите могат да сортират данни по азбучен ред, да ги филтрират и да сменят броя на елементите на една страница. Елементите са подредени в удобен за преглед вид под формата на „карти“.

3.2 Страници

3.2.1 Навигация

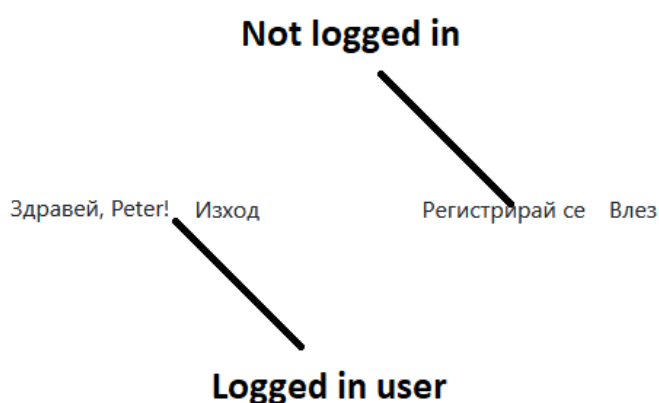
Навигацията се разполага хоризонтално в най-горната част на страницата и

остава в това положение постоянно, дори при сменяне на страниците. Линковете в нея се променят спрямо това дали има логнат потребител и каква е ролята му.



Фигура 7. Изглед от навигацията

1. Води към началната страница
2. (Само за потребители, които имат ролята „Участник“) Води към страницата с талоните на логнатия потребител
3. Води към списъка с всички тиражи
4. Води към страницата, отразяваща статистиката за числата от 1 до 49
5. Води към списъка с всички джакпоти

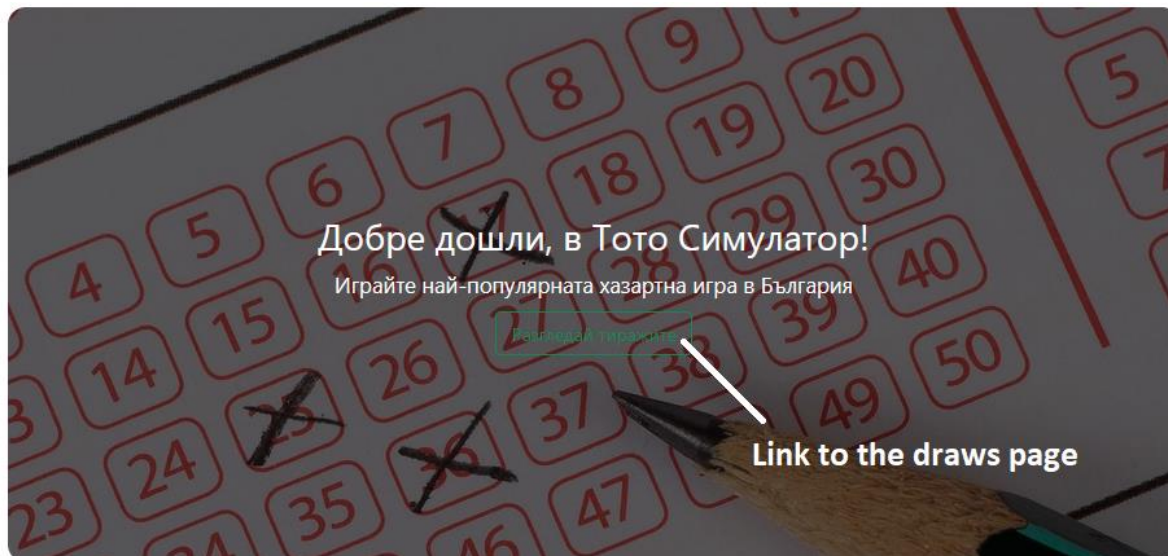


Фигура 8. Пример за промяната на навигацията при логване на потребител

Двете опции за LoginPartial частта или менюто за управление на потребителя води до страниците за регистрация и за логин, ако няма влязъл потребител и дава възможност за логанут, ако има влязъл потребител.

3.2.2 Начална страница

Началната страница е първото нещо, което потребителят вижда. Тя се състои от „hero section“ или главна секция със снимка, която представя уебсайта. В нея има заглавие, подзаглавие и зелен бутон, който води до страницата с тиражите. Под тази секция има друга, която служи за кратко обяснение на целта на сайта. Самата начална страница не е интерактивна и потребителят не може да си взаимодейства с нея.



За нас

Тото симулаторът е онлайн платформа, която предоставя на потребителите възможността да тестват късмета си, без да инвестират истински пари. Симулаторът възпроизвежда автентичните условия на тотото, като генерира случайни числа и позволява на потребителите да опитват различни комбинации и да разглеждат резултатите и печалбите си.

Създаден с цел да бъде забавен и образователен, тото симулаторът позволява на потребителите да участват в тегления без да се излагат на истински финансови загуби. Това прави платформата подходяща за всички, които се интересуват от лотарийните игри и искат да тестват възможностите на късмета си.



© 2023 - Toto_Simulator

Фигура 9. Изглед от началната страница

3.2.3 Списък с тиражи

Списъкът с тиражи представя изглед с всички тиражи до момента. Всеки потребител може да ги сортира по дата от до и по пореден номер от до. Всеки тираж

има пореден номер, дата, награден фонд (50% от всички постъпления), печеливши числа (6 напълно случайни числа) и двойки брой и натрупана сума за всяка група печеливши. 1ва група са всички талони с уцелени 6 печеливши числа, 2ра са тези, които са уцелили 5, 3та – 4 числа и 4та – 3 числа. Администраторът има възможности да изтегли нов тираж и да редактира и изтрива съществуващи такива. Бутоните за тези действия не са видими за потребители, които нямат ролята „Администратор“.

Изтегляне на тираж

Сортирай по дата

От:

До:

🔍

Сортирай по номер

От:

До:

🔍

Пореден Номер: 1012					
Дата: 13/03/2023					
Награден фонд : 2.50					
7	11	18	19	22	37
1ва група печеливши: 0	Натрупана сума: 11.00				
2ра група печеливши: 0	Натрупана сума: 0.50				
3та група печеливши: 0	Натрупана сума: 0.50				
4та група печеливши: 0	Натрупана сума: 0.50				
Редакция Изтриване					

Пореден Номер: 1011					
Дата: 13/03/2023					
Награден фонд : 5.00					
4	11	12	13	27	41
1ва група печеливши: 0	Натрупана сума: 10.00				
2ра група печеливши: 0	Натрупана сума: 1.00				
3та група печеливши: 0	Натрупана сума: 1.00				
4та група печеливши: 0	Натрупана сума: 1.00				
Редакция Изтриване					

Пореден Номер: 1010					
Дата: 13/03/2023					
Награден фонд : 0.00					
4	7	24	33	41	49
1ва група печеливши: 0	Натрупана сума: 8.00				
2ра група печеливши: 0	Натрупана сума: 0.00				
3та група печеливши: 0	Натрупана сума: 0.00				
4та група печеливши: 0	Натрупана сума: 0.00				
Редакция Изтриване					

Фигура 10. Изглед от страницата с всички тиражи

Модел на тираж:

```
public class Draw
{
    [Key]
    public int Id { get; set; }

    public DateTime Date { get; set; }

    public List<Number> Numbers { get; set; } = new List<Number>(6);

    public List<Ticket> Tickets { get; set; }
    public decimal Fund { get; set; }

    public int FirstGroupWinners { get; set; }
    public decimal FirstGroupSum { get; set; }

    public int SecondGroupWinners { get; set; }
    public decimal SecondGroupSum { get; set; }

    public int ThirdGroupWinners { get; set; }
    public decimal ThirdGroupSum { get; set; }

    public int FourthGroupWinners { get; set; }
    public decimal FourthGroupSum { get; set; }
}
```

Метода за теглене на печелившите числа:

```
public void GenerateWinningNumbers(Draw currentDraw)
{
    Random m_random = new Random();

    int[] selectedNumbers = new int[6];

    // For each array member, assign a random number
    for (int numberCount = 0; numberCount < selectedNumbers.Length;
        numberCount++)
    {
        int newNumber = m_random.Next(1, 50);

        for (int i = 0; i < numberCount; i++)
        {
            if (newNumber == selectedNumbers[i])
            {
                newNumber = m_random.Next(1, 50);
                continue;
            }
        }

        selectedNumbers[numberCount] = newNumber;
    }
    Array.Sort(selectedNumbers);

    for (int x = 0; x < selectedNumbers.Length; x++)
```

```

        {
            var number = data.Numbers.Where(n => n.Value ==
selectedNumbers[x]).FirstOrDefault();
            number.Occurrences++;
            currentDraw.Numbers.Add(number);
        }
    }
}

```

Този метод генерира напълно случайни числа, проверява дали няма повторения и след това записва финалните числа към печелившите числа на дадения тираж. Той намира съответния модел на всяко печелившо число и увеличава бройката на срещанията му. Метода се извиква в другия метод за теглене на тираж, който взема празен тираж, който е създаден при предишното извикване на метода и попълва новите данни в него. Този празен тираж е стоял невидим досега, но след като метода изчислява фондовете, проверява спечелилите, разпределя ги по групи, раздава печалби и генерира джакпот той запълва новите стойности и го прави видим за потребителите. Освен това се създава нов празен тираж, който ще бъде „изтеглен“ при следващото извикване на метода.

3.2.*Методи за редакция и изтриване

Метод за изтриване:

```

[HttpPost]
[Authorize(Roles = "Admin")]
public IActionResult Delete(int? id)
{
    Draw draw = this.data.Draws.Find(id);
    if (draw == null || draw == this.data.Draws.ToList().OrderByDescending(d =>
d.Date).FirstOrDefault())
    {
        return BadRequest();
    }

    var drawTickets = this.data.Tickets.Where(t => t.DrawId == draw.Id);
    if (drawTickets.Any())
    {
        foreach (var ticket in drawTickets)
        {
            this.data.Tickets.Remove(ticket);
        }
    }

    this.data.Draws.Remove(draw);
    this.data.SaveChanges();
    return RedirectToAction("All", "Draws");
}

```

Пример за метод за изтриване. Подобни методи за използвани за изтриване на всеки един модел от приложението. Методът намира дадения елемент по “id” и

го изтрива от базата данни. В дадения пример, методът изтрива също така всеки талон, който е попълнен за дадения тираж. Накрая промените се запазват и методът препраща потребителя отново към дадената каталог страница.

Метод за редакция:

```
[HttpPost]
[Authorize(Roles = "Admin")]
public IActionResult Edit(int id, DrawFormModel drawModel)
{
    var draw = this.data.Draws.Where(d => d.Id == id).FirstOrDefault();
    if (draw == null)
    {
        return BadRequest();
    }

    draw.Fund = drawModel.Fund;
    draw.FirstGroupWinners = drawModel.FirstGroupWinners;
    draw.FirstGroupSum = drawModel.FirstGroupSum;
    draw.SecondGroupWinners = drawModel.SecondGroupWinners;
    draw.SecondGroupSum = drawModel.SecondGroupSum;
    draw.ThirdGroupWinners = drawModel.ThirdGroupWinners;
    draw.ThirdGroupSum = drawModel.ThirdGroupSum;
    draw.FourthGroupWinners = drawModel.FourthGroupWinners;
    draw.FourthGroupSum = drawModel.FourthGroupSum;

    this.data.SaveChanges();
    return RedirectToAction("All", "Draws");
}
```

Пример за метод за редакция. Подобни методи за използвани за редакция на всеки един модел от приложението. Методът намира дадения елемент по “id” и задава новите стойности на записа му в базата данни. Този метод се извиква от форма за редакция, когато се направи “post” заявка. Накрая промените се запазват и методът препраща потребителя отново към дадената каталог страница.

Редакция на тираж

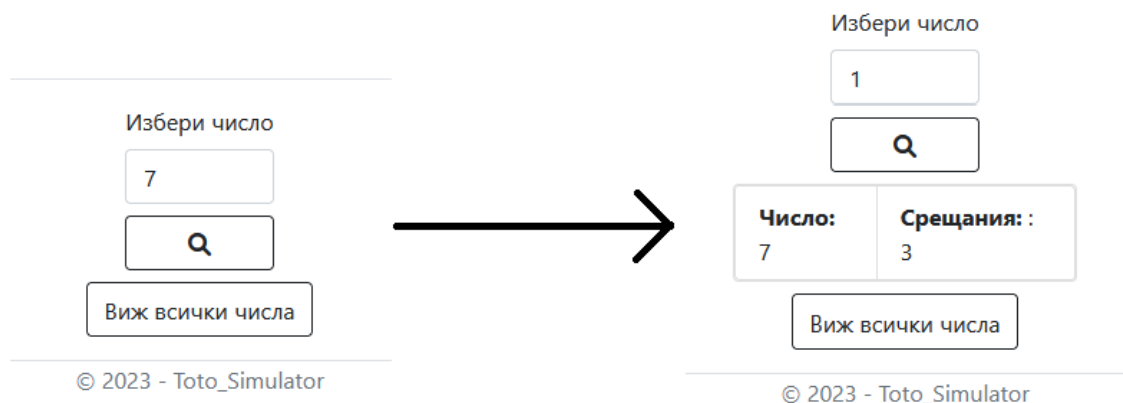
Награден фонд	<input type="text" value="5.00"/>
Победители от първа група	<input type="text" value="0"/>
Награден фонд за първа група	<input type="text" value="10.00"/>
Победители от втора група	<input type="text" value="0"/>
Награден фонд за втора група	<input type="text" value="1.00"/>
Победители от трета група	<input type="text" value="0"/>
Награден фонд за трета група	<input type="text" value="1.00"/>
Победители от четвърта група	<input type="text" value="0"/>
Награден фонд за четвърта група	<input type="text" value="1.00"/>

Редактирай

Фигура 11. Пример за форма за редакция от приложението

3.2.4 Талони

В приложението има два каталога за талони. Единият достъпен само от потребители с ролята „Участник“, а другият само от администратори. Каталогът за участници има съдържа талони само на логнатия потребител. Той може да ги сортира по номер, да избира броя на елементите на страница(5, 10 или 15) и също така да вижда печалбите си. Най-отдолу на страницата има избор за следваща и предишна страница.



Фигура 13. Изглед от страницата за избиране на числа

Другата страница от секцията включва изглед на всички числа от 1 до 49 и броя на срещанията на всяко едно.

Избери число											
Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :
1	0	2	2	3	2	4	6	5	3		
Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :
6	0	7	3	8	3	9	1	10	1		
Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :
11	4	12	2	13	2	14	2	15	1		
Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :
16	1	17	2	18	2	19	1	20	1		
Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :
21	2	22	2	23	0	24	2	25	1		
Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :
26	3	27	3	28	1	29	1	30	1		
Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :	Число:	Срещания: :

Фигура 14. Изглед от страницата с всички числа

3.2.6 Джакпоти

Страницата с джакпотите извежда изглед на всички джакпоти в базата. Нов джакпот се създава винаги, когато се тегли тираж. Опциите за редактиране и изтриване са достъпни само за администратори. Всеки потребител може да сортира джакпотите по пореден номер или по сума.

Сортирай по номер

Номер на джекпот: 2	Натрупана сума: : 10000.00
Редакция	Изтриване

Номер на джекпот: 6	Натрупана сума: : 10005.00
Редакция	Изтриване

Номер на джекпот: 9	Натрупана сума: : 10005.60
Редакция	Изтриване

Номер на джекпот: 12	Натрупана сума: : 0.00
Редакция	Изтриване

Номер на джекпот: 15	Натрупана сума: : 0.00
Редакция	Изтриване

Номер на джекпот: 18	Натрупана сума: : 8.00
Редакция	Изтриване

Сортирай по сума

Номер на джекпот: 4	Натрупана сума: : 10000.00
Редакция	Изтриване

Номер на джекпот: 7	Натрупана сума: : 10005.00
Редакция	Изтриване

Номер на джекпот: 10	Натрупана сума: : 10005.60
Редакция	Изтриване

Номер на джекпот: 13	Натрупана сума: : 0.00
Редакция	Изтриване

Номер на джекпот: 16	Натрупана сума: : 2.00
Редакция	Изтриване

Номер на джекпот: 19	Натрупана сума: : 8.00
Редакция	Изтриване

Номер на джекпот: 5	Натрупана сума: : 10003.60
Редакция	Изтриване

Номер на джекпот: 8	Натрупана сума: : 10005.60
Редакция	Изтриване

Номер на джекпот: 11	Натрупана сума: : 0.00
Редакция	Изтриване

Номер на джекпот: 14	Натрупана сума: : 0.00
Редакция	Изтриване

Номер на джекпот: 17	Натрупана сума: : 7.00
Редакция	Изтриване

Номер на джекпот: 1007	Натрупана сума: : 8.00
Редакция	Изтриване

Фигура 15. Изглед от страницата с всички джекпоти

3.2.6.* Контролер на джекпотите и IActionResult за Index страница

Метода, който визуализира джекпотите е много подобен на всички останали методи, които връщат страници тип „каталог“. Той включва и сортировките в себе си.

```
public IActionResult Index(string sortOrder)
{
    ViewData["IdSortParam"] = string.IsNullOrEmpty(sortOrder) ? "id_desc"
: "";
    ViewData["SumSortParam"] = sortOrder == "Sum" ? "sum_desc" : "Sum";

    IQueryable<Jackpot> jackpots = from j in this.data.Jackpots select j;
    switch (sortOrder)
    {
        case "id_desc":
            jackpots = jackpots.OrderByDescending(s => s.Id);
            break;
        case "Sum":
            jackpots = jackpots.OrderBy(s => s.AccumulatedSum);
            break;
        case "sum_desc":
            jackpots = jackpots.OrderByDescending(s =>
s.AccumulatedSum);
            break;
    }
}
```

```

        default:
            jackpots = jackpots.OrderBy(s => s.Id);
            break;
    }

    return View(jackpots);
}

```

Останалите методи за визуализация на каталог са подобни, но някои добавят в себе си различни видове сортировки, филтрации, търсене или странициране.

3.2.7 Администраторска страница (Потребители)

Администраторската страница за потребителите дава информация за всеки регистриран потребител в базата в табличен вид и дава възможности за контрол на администраторите. Те могат да редактират информацията за потребителя, да го изтрият от базата или да му сменят ролята. Смяната на ролята става чрез „toggle” бутон „Направи Администратор” или „Направи Участник” спрямо ролята на избрания потребител. Администраторите могат да търсят потребители по техните имена или фамилии, да ги сортират по азбучен ред на имената и да променят броя на потребителите, които да се показват на една страница (5, 10 или 15).

Имя	Потребителско имя	E-Mail	Действия	Печалба
Ivo Grigorov	Ivo	ivo@gmail.com	Редакция Направи Администратор Изтриване	10.00
John Ignatov	John	john@abv.bg	Редакция Направи Администратор Изтриване	14.92
Peter Jackson	Peter	peter@abv.bg	Редакция Направи Участник Изтриване	10.00

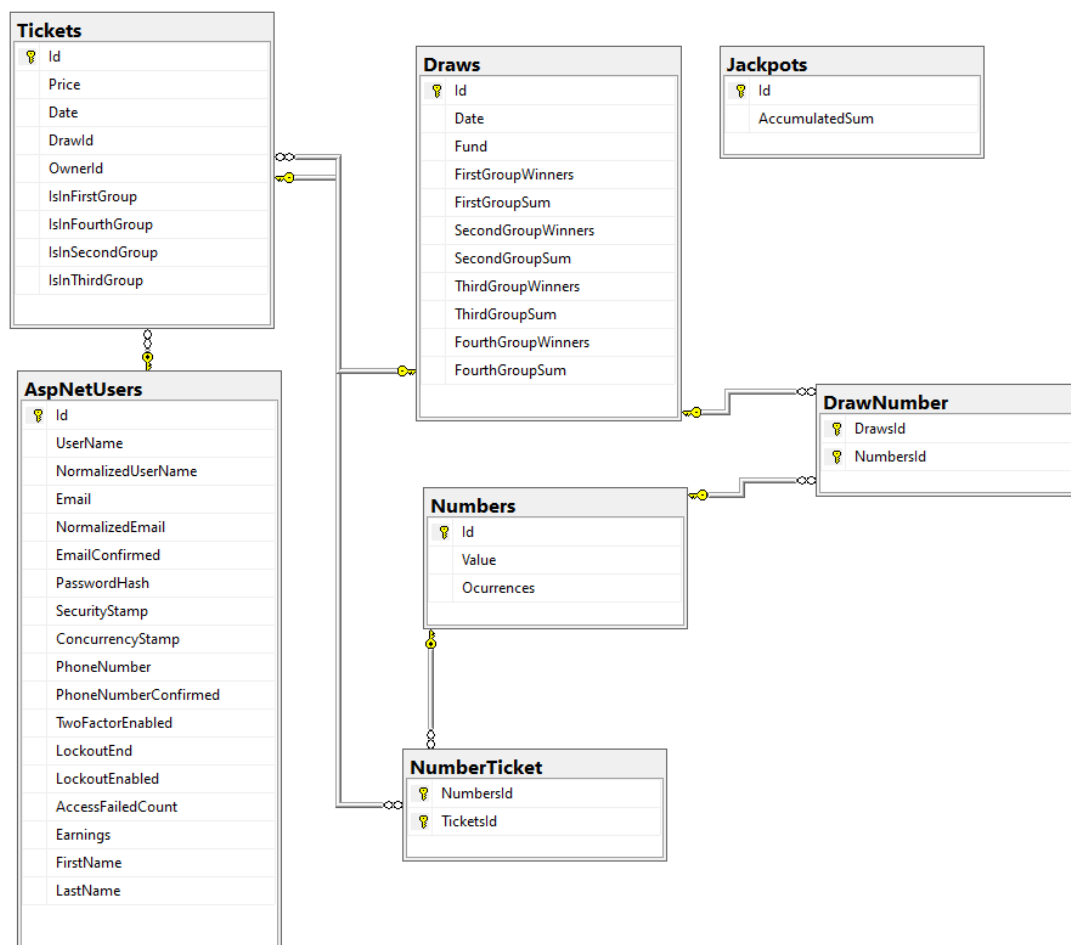
© 2023 - Toto_Simulator

Фигура 16. Изглед от страницата с всички потребители

Това е единствената уникална страница, която е достъпна единствено от администраторите(заедно с формата за редактиране на потребител). Останалите страници са почти идентични за всички потребители, но за администраторите се дават повече възможности или информация.

3.3 Модели и база данни

Приложението се състои от няколко модела, които изграждат структурата на тотото. Това са : талони, тиражи, числа, джакпоти и потребители. Моделите имат връзки помежду си. Талоните имат един собственик(потребител), потребителите имат много талони, талоните имат един тираж, докато тиражите имат много участващи билети. Това са връзките от тип „едно към много“. Освен тях има и връзки от тип „много към много“ талоните и тиражите имат много числа, докато едно число може да присъства в много талони и тиражи.



Фигура 17. Графика представяща схемата на моделите в базата данни изграждаща приложението

4. ЗАКЛЮЧЕНИЕ

Проектът „ТОТО СИМУЛАТОР“ е уеб приложение разработено с ASP.NET, C#, HTML, CSS в MVC архитектурата. То има целта да симулира теглене на числа , натрупване на джакпот, разпределение на печалби по групи, статистика, отразяване на печалби на потребител. Уеб приложението си комуникира с база данни, като поддържа всички CRUD операции. Системата поддържа роли „Администратор“ и „Участник“ всяка с различни права. Участниците могат да попълват талони и да участват в тегленията с цел печалба, която ще бъде отразена в профила им. Тегленето се извършва автоматично, след натискането на бутон от администраторите, заедно с натрупването на джакпот, разпределението на сумите по групи и раздаването на индивидуални печалби.

Изгледите с информация за талоните, тиражите, числата, джакпотите и потребителите са подредени по удобен за преглед начин, както и с възможности за филтрация, сортиране, търсене и странициране, което допринася за постигането на задоволителен UX (user experience). Освен удобни изгледи, приложението дава богат контрол на администраторите.

Приложението има много възможности за бъдещо развитие и потенциална комерсиализация. То може да се разрастне чрез добавяне на допълнителни статистики и нови видове тиражи и билети. Както в много лотарийни игри може да се добавят „втори шансове“, бонус числа за по-скъпи талони, допълнителни тегления и др. Проектът е изпълнил всички изисквания по условие, но предполага развитие в бъдеще с още по-богати функционалности.

Приложението служи като забавна онлайн игра с некомерсиален характер, която може да образова потребителите си за това как работи една лотарийна игра, без да създава риск от загуба на парични средства при участие.

Използвана литература

"A Complete Guide to Flexbox" от CSS-Tricks - <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

"A Complete Guide to Grid" от CSS-Tricks - <https://css-tricks.com/snippets/css/complete-guide-grid/>

"CSS-Tricks" - <https://css-tricks.com/>

"MDN Web Docs - раздел за CSS - " - <https://developer.mozilla.org/en-US/docs/Web/CSS>

"Responsive Web Design" от Ethan Marcotte - <https://alistapart.com/article/responsive-web-design/>

"W3Schools - раздел за CSS " - <https://www.w3schools.com/css/>

GeeksforGeeks - Model-View-Controller(MVC) architecture:
<https://www.geeksforgeeks.org/model-view-controllermvc-architecture-for-node-applications/>

HTML5 Rocks - семантични елементи -
<https://www.html5rocks.com/en/features/semantics>

<https://bg.wikipedia.org/wiki/%D0%9B%D0%BE%D1%82%D0%B0%D1%80%D0%B8%D1%8F>

<https://en.wikipedia.org/wiki/Database>

https://en.wikipedia.org/wiki/Database_design

https://en.wikipedia.org/wiki/Database_management_system

<https://www.databasestar.com/database-design-tutorial/>

https://www.tutorialspoint.com/database_design/index.htm

Microsoft Docs - ASP.NET Core MVC: <https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0>

Microsoft Learn – Увод в Entity Framework Core: <https://docs.microsoft.com/en-us/learn/modules/introduction-to-entity-framework-core/>

Microsoft SQL Server Management Studio (SSMS):

Microsoft SQL Server:

Mozilla Developer Network (MDN) - раздел за HTML -
<https://developer.mozilla.org/en-US/docs/Web/HTML>

Tutorialsteacher - MVC Architecture: <https://www.tutorialsteacher.com/mvc/mvc-architecture>

W3Schools - раздел за HTML - <https://www.w3schools.com/html/>

W3Schools - раздел за SQL - <https://www.w3schools.com/sql/>

Wikipedia - Model–view–controller:
<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

Документация на Microsoft SQL Server - <https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-ver15>

Официална документация на Entity Framework Core: <https://docs.microsoft.com/en-us/ef/core/>

Официална страница на Microsoft SQL Server - <https://www.microsoft.com/en-us/sql-server/>

Официална страница на SSMS - <https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15>

Официалната документация на HTML5 от W3C - <https://www.w3.org/TR/html52/>