|  |  |  |
| --- | --- | --- |
| **МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА** | | |
| **ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО ЕЛЕКТРОТЕХНИКА И ЕЛЕКТРОНИКА**  бул. Пещерско шосе № 26  4002 гр. Пловдив, България  тел. 032 / 643-657  info-1690174@edu.mon.bg |  | **VOCATIONAL SCHOOL OF ELECTRICAL ENGINEERING AND ELECTRONICS**  26 Peshtersko chaussee blvd.,  4002 Plovdiv, Bulgaria  Phone: 032 / 643-657  info-1690174@edu.mon.bg |

**професия код 481030 „Приложен програмист“**

**специалност код 4810301 „Приложно програмиране“**

**ДИПЛОМЕН ПРОЕКТ**

**за придобиване трета степен професионална квалификация**

**Тема: HFit – сайт за тренировки в къщи**

**Дипломант**: …………… **Ръководител-консултант**: …………

/Илиян Куршумов/ /инж. Магдалена Маглижанова/

Клас: 12а

e-mail: ilijan.kurshumov@gmail.com

Пловдив

2025 година

# Съдържание

[Увод 2](#_Toc183427329)

[Глава 1 ……….. 3](#_Toc183427330)

[1.1. Заглавие на параграф 3](#_Toc183427331)

[1.2. Заглавие на параграф 3](#_Toc183427332)

[1.3. Заглавие на параграф 3](#_Toc183427333)

[1.4. Заглавие на параграф 3](#_Toc183427334)

[1.5. Заглавие на параграф 3](#_Toc183427335)

[Глава 2 ………. 4](#_Toc183427336)

[2.1. Заглавие на параграф 4](#_Toc183427337)

[2.2. Заглавие на параграф 4](#_Toc183427338)

[2.3. Заглавие на параграф 4](#_Toc183427339)

[2.4. Заглавие на параграф 4](#_Toc183427340)

[2.5. Заглавие на параграф 4](#_Toc183427341)

[Глава 3 ………………….. 5](#_Toc183427342)

[3.1. Заглавие на параграф 5](#_Toc183427343)

[3.2. Заглавие на параграф 5](#_Toc183427344)

[3.3. Заглавие на параграф 5](#_Toc183427345)

[3.4. Заглавие на параграф 5](#_Toc183427346)

[3.5. Заглавие на параграф 5](#_Toc183427347)

[Глава 4 …………… 6](#_Toc183427348)

[4.1. Заглавие на параграф 6](#_Toc183427349)

[4.2. Заглавие на параграф 6](#_Toc183427350)

[4.3. Заглавие на параграф 6](#_Toc183427351)

[4.4. Заглавие на параграф 6](#_Toc183427352)

[4.5. Заглавие на параграф 7](#_Toc183427353)

[Заключение 8](#_Toc183427354)

[Използвани Източници 9](#_Toc183427355)

[Приложение 1 10](#_Toc183427356)

[Приложение 2 11](#_Toc183427357)

[Приложение 3 12](#_Toc183427358)

[Приложение 4 13](#_Toc183427359)

# Увод

В съвременния свят, където времето е ценен ресурс, а виртуалното пространство играе все по-голяма роля в ежедневието, нуждата от удобни и ефективни решения за поддържане на здравословен начин на живот нараства. HFit е иновативна платформа, създадена с мисията да направи тренировките у дома достъпни, лесни и мотивиращи за хората, които искат да се грижат за своето физическо и психическо здраве, без да напускат комфорта на дома си.

HFit предоставя на потребителите персонализирани тренировъчни програми, интерактивни уроци, следене на напредъка и богата база данни от упражнения, разделени по категории като кардио, силови тренировки и гъвкавост. Сайтът е създаден, за да отговори на нуждите както на начинаещите, така и на напредналите фитнес ентусиасти, които търсят ефективен начин да постигнат своите цели.

Основната цел на проекта е да се изгради интуитивна и многофункционална уеб платформа, която не само да улеснява тренировките у дома, но и да мотивира потребителите да водят по-здравословен начин на живот. Платформата включва възможности за следене на индивидуалния прогрес, достъп до инструкции за упражнения и калкулоране на калории.

HFit е за всеки, който иска да подобри здравето си, но няма нужното време за отделяне за да ходи на фитнес. Няма занчение от пола, мъж или жена, всеки е добре дошъл в семейството на HFit.

Очакваният резултат е напълно функционираща уеб платформа, която предоставя удобство, мотивация и подкрепа на потребителите в тяхното фитнес пътешествие. HFit не е просто инструмент за тренировки – това е онлайн партньор, която вдъхновява и насърчава хората да постигат целите си и да подобряват качеството на своя живот.

В първа глава на разработката се разглеждат теоретичните аспекти, на които тя е базирана. Осъществен е преглед на основни технологични подходи, използвани за създаването на подобни разработки, както и технологиите, които биват използвани в крайния продукт.

Във втора глава на разработката е разкрита идеята зад работата на приложението. В нея се намира информация за това по какъв начин са свързани отделните технологии и процесите, които те извършват, за да функционира приложението. Освен това, там е разкрита и архитектурата на приложението заедно с всички файлове, намиращи се в него.

Трета глава на разработката разкрива работата на приложението. В нея се намират извадки от кода на някои от основните функционалности и обясненията зад това как работят те.

В четвърта глава на разработката е разкрито поведението на приложението и отделните му функции. В нея се намират изображения, показващи визуализацията на разработката, както и разяснение за работата с нея.

# Глава 1 - ТЕОРЕТИЧНИ ОСНОВИ И КОНЦЕПЦИИ ПО ТЕМАТА

## 1.1. Обзор на технологиите

В съвременния свят разработчиците имат богат избор от среди за разработка, които улесняват и оптимизират процеса на създаване на софтуерни приложения. За стартиране на даден проект е важно да се избере подходяща среда, която да съответства на нуждите и целите на разработката. За разработката на приложението „**HFit**“ е използвана **Visual Studio Code** – лек и мощен текстов редактор, който предоставя широк спектър от инструменти и разширения. Изборът на тази среда е мотивиран от нейната гъвкавост, лесна интеграция с различни технологии и богатата екосистема от плъгини.

При проектирането на настоящата разработка е извършен подробен анализ на възможните софтуерни технологии, целящи да осигурят ефективно и иновативно решение. Фокусът падна върху следните технологии: **Next.js, TypeScript, Firebase, AWS S3, React, CSS** и **HTML.**

* + 1. **Next.js**

Next.js е React-базирана рамка, която предлага възможности за изграждане на съвременни уеб приложения, като осигурява сървърно-рендериране (SSR) и статично генериране (SSG), което подобрява производителността и SEO оптимизацията. Тя играе ключова роля в проектите, като позволява динамично зареждане на страници и ускорява разработката благодарение на лесната структура на файловете и автоматично маршрутизиране.

Next.js включва и поддръжка за API маршрути, което улеснява интегрирането на бекенд функционалности в рамките на същото приложение. Сред основните предимства на рамката са не само високата производителност и бързина, но и лесната интеграция с TypeScript и други инструменти. Next.js предлага множество готови оптимизации за производителност, като автоматично разделяне на кода и оптимизиране на изображенията, което прави изграждането на уеб приложения още по-ефективно.[[1]](#footnote-1)

* + 1. **Typescript**

TypeScript е език за програмиране, който разширява JavaScript чрез добавяне на статично типизиране. Той играе важна роля в разработката на приложения, като помага за предотвратяване на грешки по време на писането на код и осигурява по-добра поддръжка на кода в дългосрочен план. В проекта TypeScript гарантира правилното използване на функции и променливи, като осигурява типизация на данните.

Това подобрява четимостта и разбирането на кода, като намалява риска от грешки, свързани с типове. Сред основните предимства на TypeScript са по-добрата предвидимост и безопасност при писането на код, лесното откриване на грешки по време на разработка и по-добрата интеграция с IDE, като Visual Studio Code, което води до интелигентно автодовършване и рефакториране на кода.[[2]](#footnote-2)

* + 1. **Firebase**

Firebase, на Фигура 1 е показано логото на технологията, е облачна платформа, разработена от Google, която предоставя широка гама от услуги за уеб и мобилни приложения, включително бази данни, автентикация, хостинг, анализи, облачни функции и други. Тя е проектирана да улесни разработчиците в създаването на надеждни и мащабируеми приложения, като същевременно намали сложността при управлението на сървърите и инфраструктурата.

Едно от най-големите предимства на Firebase е неговата лесна интеграция с Next.js, което прави платформата изключително удобна за разработчици, работещи с модерни уеб технологии. Освен това, Firebase осигурява висока степен на сигурност на данните, мащабируемост и надеждност. Платформата предлага и готови решения за автентикация чрез различни доставчици, което значително ускорява разработката и позволява на екипите да се фокусират върху основните функционалности на приложението.[[3]](#footnote-3)



Фигура 1 – Лого на Firebase

* + 1. **AWS S3**

Amazon Simple Storage Service (S3) е облачна услуга за съхранение на файлове, предоставена от AWS, която осигурява висока надеждност и достъпност при съхранение и извличане на данни. В проекта AWS S3 се използва за съхранение на изображения, свързани с тренировките, които са качени от администраторите.

Тя осигурява мащабируемо решение за съхранение на медийни файлове и замества Firebase Storage, предоставяйки по-голяма гъвкавост и съвместимост с други AWS услуги. Сред основните предимства на S3 са лесното управление на големи обеми данни, висока производителност и надеждност, както и възможността за интеграция с други AWS услуги, като CloudFront, за ускоряване на доставката на съдържание.[[4]](#footnote-4)

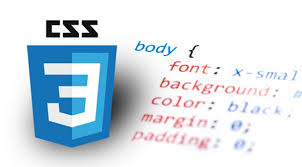
* + 1. **React**

React е библиотека за създаване на потребителски интерфейси, разработена от Meta, и е основата на Next.js. Тя е широко използвана за разработка на уеб приложения благодарение на своите предимства. В проекта React се използва за изграждане на компонентен интерфейс, състоящ се от повторно използваеми елементи, което осигурява бърза и интерактивна работа с потребителския интерфейс. React също така позволява ефективно управление на състоянието на компонентите с помощта на React Hooks.

Сред основните предимства на React са гъвкавостта и лесното управление на състоянието на приложението, възможността за създаване на сложни потребителски интерфейси и силната общност, подкрепена от богата екосистема от библиотеки и инструменти.[[5]](#footnote-5)

* + 1. **CSS**

CSS (Cascading Style Sheets) служи като език за стилизиране на HTML елементи. В проекта той играе ключова роля, като осигурява основното визуално оформление на компонентите и страниците. CSS позволява персонализиране на външния вид на приложението, като го съобразява с тематиката на проекта.

 Сред основните му предимства се открояват гъвкавостта, пълният контрол върху стилизирането, съвместимостта с всички уеб браузъри и лесната интеграция с Next.js. Въпреки някои ограничения, като липсата на капсулация и ограничените изчислителни възможности, CSS остава изключително ефективен инструмент за стилизиране на уеб приложения.[[6]](#footnote-6) На Фигура 2 е показано логото на технологията.

Фигура 2 - Лого на CSS

* + 1. **HTML**

HTML (HyperText Markup Language) е основният език за създаване на структурата и съдържанието на уеб страници (Фигура 3). Той осигурява семантична основа за всички елементи и улеснява стилизирането и подредбата с CSS. Основните му предимства са простотата, универсалността и съвместимостта с всички браузъри.[[7]](#footnote-7) На Фигура 2 е показано логото на технологията.



Фигура 3 - Лого на HTML

# Глава 2 - Структура на Разработката

## 2.1. Обща схема

Общата структура на разработката е изработена чрез Next.js. Това е full-stack Javascript framework, който позволява разработката както на front-end(клиентска част), така и на back-end(сървърна част).На фигура 4 е показана връзката между тези части и другите елементи от проекта.

A diagram of a software development

Description automatically generated

Фигура 4 - Структура на проекта

## 2.2. Какво е front-end (клиентска част)

Клиентската (Frontend) част в Next.js е съществена за създаването на динамични и бързи потребителски интерфейси. Като рамка, изградена върху React, Next.js предоставя всички необходими инструменти за разработка на модерни уеб приложения. Клиентската част обхваща всичко, което потребителят вижда и с което взаимодейства в браузъра.

Основните характеристики на клиентската част в Next.js включват използването на React компоненти за изграждане на потребителския интерфейс, автоматична оптимизация на страниците чрез Static Site Generation (SSG) за по-бързо зареждане, лесна навигация чрез вградената система за маршрутизация и поддръжка на Client-Side Rendering (CSR) за динамични уеб приложения.

## 2.3. Какво е back-end (сървърна част)

Сървърната (Backend) част в Next.js предоставя мощни функционалности, които позволяват обработка на заявки, взаимодействие с база данни и динамично рендиране на съдържание. Освен че се грижи за клиентската част, Next.js разполага с вградени инструменти за създаване на сървърни API маршрути и подобряване на производителността на уеб приложенията.

Основните характеристики на сървърната част в Next.js включват API маршрути, които позволяват създаване на бекенд ендпойнти в самото приложение за обработка на HTTP заявки (GET, POST, PUT, DELETE). Също така, сървърното рендиране (Server-Side Rendering – SSR) осигурява извличане на данни на сървъра и изпращане на напълно рендирана страница към клиента, което подобрява SEO и производителността. Освен това, Next.js позволява интеграция с бази данни, което дава възможност за обработка на динамични заявки и съхраняване на информация.[[8]](#footnote-8)

## 2.4. Структута на Базата данни

Firebase в нерелационна база от данни, тя използва колекции, а не таблици за да запазва данните. Колекциите са контейнери за документи, които съхраняват данни. Те представляват основен начин за организиране на информацията в базата данни.[[9]](#footnote-9) В разработката изграждаме 3 колекции, всяка с нейните отделни полета.

Първата колекция е "Users", тя съдържа шест полета:"id", "usename","email","password", "gender" и "role". Тя служи за запазването на информацията на потребителя, за регистрирането и влизането в акаунт.

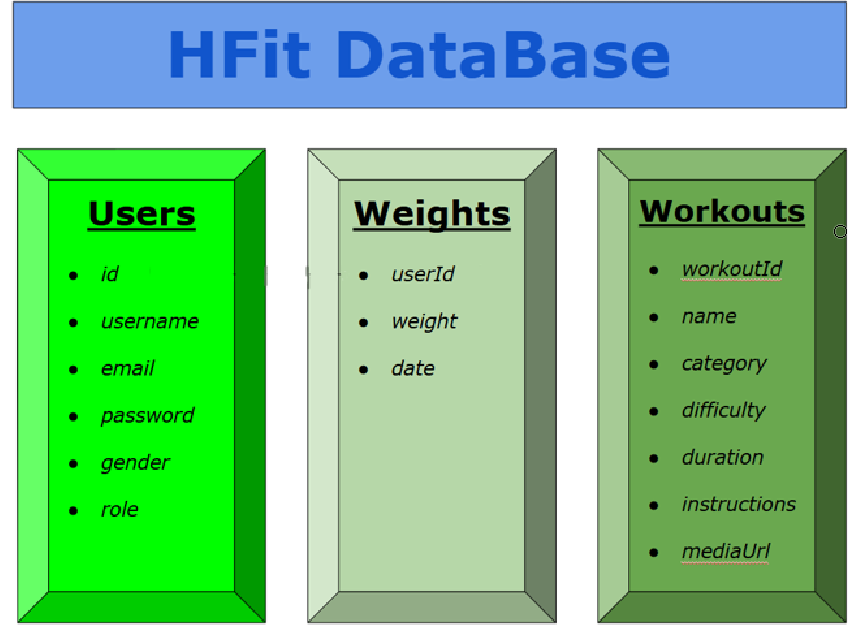
Втората колекция е "Weights", тя има три полета: "userId", "weight", "date". Всеки един потребител може да записва теглото си и то е запазено заедно с датата, на която е въведено. На Фигура 5 е показан примерен запис в базазта данни.

A screenshot of a computer

AI-generated content may be incorrect.

Фигура 5 - Запис в колекцията “Weights”

Третата колекция е "Workouts", тя запазва информацията за създаване на нова тренировка от административния профил, също така показва и записаните тренировки. Колекцията има седем полета, те са: "workoutId", "name", "category", "difficulty", "duration", "instructions" и "mediaUrl", чрез това поле се запазва адреса на изображенията на тренировките в AWS S3. В изображение 6 е показана структурата на колекциите.



Фигура 6 - Структура на колекциите

## 2.5. Структурата на файловете в разработката

Основната част на приложението се намира в папката **app/**. Тя съдържа всички ключови страници, компоненти и логически модули, необходими за работата на платформата. Вътре се намират няколко важни директории, всяка със своето предназначение. Показано на фигура 7.

A screenshot of a computer

AI-generated content may be incorrect.

Фигура 7 – Структура на папка „/app“

Папката **Amazon S3/** съхранява конфигурационния фай и логиката, свързана с **AWS S3**, което позволява качване и управление на изображения за тренировките. Това осигурява ефективно и мащабируемо решение за съхранение на мултимедийно съдържание.

В **calorie\_calculator/** се намират компонентите и логиката за изчисляване на дневния калориен прием според въведените от потребителя данни. Резултатите от тези изчисления се визуализират в отделната папка **calorie\_calculator\_results/**, което позволява по-добро разделение на кода и организацията на функционалността.

Функционалността за управление на тренировъчните сесии се намира в **training\_sessions/**. Тук админът може да добавя, редактира и преглежда тренировъчните програми, коите потребителите могат да изпълняват. От друга страна, папката **weight\_progress/** съдържа логиката за проследяване на теглото във времето, което позволява на потребителите да наблюдават напредъка си чрез визуални графики.

За обработката на автентикацията са създадени отделни папки – **login/** и **register/**. Автентикацията става или чрез въвеждане на информация във форм, или чрез влизане с Google акаунт. В тях са разположени файловете за вход и регистрация на потребителите, което прави кода по-структуриран и четим. След успешен вход, приложението показва персонализирана информация в навигационната лента, включително името и профилната снимка на потребителя.

Папката **components/** съдържа всички споделени React компоненти, използвани в различни части на приложението. Тези компоненти помагат за поддържането на чист и модулен код, като намаляват повторението на една и съща функционалност в различни файлове.

A screenshot of a computer program

AI-generated content may be incorrect.

Фигура 8 - Компоненти в папката „/components“

A screen shot of a computer

AI-generated content may be incorrect.Друга важна част от проекта е **firebase/**, която отговаря за интеграцията с Firebase. Тук се намират файловете, отговарящи за автентикацията на потребителите, както и за взаимодействието с базата данни Firestore. Чрез Firebase се съхраняват и извличат данни, като например информация за тренировъчните сесии и теглото на потребителите.

Фигура 9 - Файлове за конфигуриране на FireBase

**public/** е директория за статични ресурси като изображения, икони и други файлове, които не се променят динамично. В тази папка се съхраняват всички мултимедийни елементи, които трябва да бъдат достъпни директно в браузъра.

**styles/** съдържа глобалните стилове на приложението, включително **globals.css**, който дефинира основния външен вид на уеб страницата. В комбинация с **Tailwind CSS**, проектът използва модерни методи за стилизиране, които позволяват бързо и ефективно изграждане на дизайна.

Сред най-важните конфигурационни файлове са (Цялата файлова структура може да се рагледа на Фигура 10):

* **layout.tsx** – основен компонент, който задава структурата на всички страници.
* **page.tsx** – начална страница на приложението.
* **.env.local** – файл, съдържащ чувствителни данни като API ключове и конфигурации (не се качва в GitHub).
* **.eslintrc.json** – конфигурация за ESLint, който помага да се поддържа добър стил на кода.
* **.gitignore** – файл, определящ кои файлове и папки да бъдат игнорирани при качване в Git.
* **next.config.mjs** – основен конфигурационен файл за Next.js, който управлява различни настройки на приложението.
* **package.json** – описва зависимостите на проекта, както и наличните команди за разработка.
* **postcss.config.js** и **postcss.config.mjs** – конфигурационни файлове за PostCSS, който се използва за трансформиране на CSS.
* **tailwind.config.js** – конфигурация за Tailwind CSS, определяща персонализирани цветове, шрифтове и други стилови настройки.
* **tsconfig.json** – конфигурация за TypeScript, която указва как трябва да се компилират TypeScript файловете в проекта.

A screenshot of a computer

AI-generated content may be incorrect.

Фигура 10 – Файлова Структура

# Глава 3 - ПРАКТИЧЕСКА РЕАЛИЗАЦИЯ НА РАЗРАБОТКАТА

## 3.1. Login / Registration система

Ще започнем с разглеждането на метода „**registerWithEmail**“. Този метод приема параментрите дадени от потребителя и регистрира нов потребител с имейл и парола в Firebase Authentication. Тук също се проверява дали потребителя, който иска да се регистрира, е администратор, чрез полето “role”, което по презумция е запазено като „user“. Ако потребителя се регистрира с администраторския имей това поле се променя на „admin“. В кода по-долу е демонстриран кода на функцията.

export const registerWithEmail = async (email: string, password: string, username: string, gender: string, role: string = "user") => {

  const userCredential = await createUserWithEmailAndPassword(auth, email, password);

  await updateProfile(userCredential.user, { displayName: username });

  const isAdmin = email === "ilijan.kurshumov@gmail.com";

  const userRole = isAdmin ? "admin" : role;

  await setDoc(doc(db, 'users', userCredential.user.uid), {

    username,

    email,

    gender,

    createdAt: new Date(),

    role: userRole,

  });

  return userCredential.user;

};

Методът „**loginWithUsernameOrEmail**“ позволява на потребителя да влезе в профила си с потребителско име или имейл и парола. За да разберем кое от двете е въвел потребителя се извършва проверка на въведената стойност, дали съдържа силволът “@”. Ако има този символ то потребителя е въвел имейл. След като сме разбрали кое от двете е въведено, и ако информацията е правилна, се извършва заявка към Firestore за намиране на имейла или името на потребителя. По-долу е демонстриран кода на функцията.

export const loginWithUsernameOrEmail = async (identifier: string, password: string) => {

  let email = identifier;

  if (!identifier.includes('@')) {

    const usersRef = collection(db, 'users');

    const q = query(usersRef, where('username', '==', identifier));

    const querySnapshot = await getDocs(q);

    if (querySnapshot.empty) {

      throw new Error("Username not found");

    }

    email = querySnapshot.docs[0].data().email;

  }

  return signInWithEmailAndPassword(auth, email, password);

};

Методът „**signInWithGoogle**“ Позволява на потребителя да влезе чрез Google акаунт. Първоначално се опитва вход чрез popup за по-добро потребителско изживяване. Ако popup методът не работи (например при блокиране от браузъра), преминава към redirect вход. След успешен вход запазва или актуализира потребителя в Firestore. Кода за функцияте е представен долу. Тези методи, заедно със спомагателни методи за регистриране и влизане в профили, се намират в Приложение 1.

export const signInWithGoogle = async () => {

  try {

    localStorage.removeItem('googleAuthError');

    try {

      const result = await signInWithPopup(auth, googleProvider, browserPopupRedirectResolver);

      await saveGoogleUserToFirestore(result.user);

      return result.user;

    } catch (error:any) {

      if (error.code === 'auth/popup-blocked' ||

          error.code === 'auth/popup-closed-by-user' ||

          error.code === 'auth/cancelled-popup-request') {

        localStorage.setItem('googleAuthStartTime', Date.now().toString());

        await signInWithRedirect(auth, googleProvider);

        return null;

      }

      throw error;

    }

  } catch (error:any) {

    localStorage.setItem('googleAuthError', error.message || 'Unknown error');

    throw error;

  }

};

## 3.2. Заглавие на параграф

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

## 3.3. Заглавие на параграф

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

## 3.4. Заглавие на параграф

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

## 3.5. Заглавие на параграф

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

# Глава 4 ……………

## 4.1. Заглавие на параграф

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

## 4.2. Заглавие на параграф

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

## 4.3. Заглавие на параграф

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

## 4.4. Заглавие на параграф

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

## 4.5. Заглавие на параграф

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

# Заключение

Съдържание на заключение. …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

Съдържание заключение …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

# Използвани Източници

[Бек, Д. & Бек](https://www.orangecenter.bg/author/dzhesika-nyuman-bek-mat-bek.html), М. (2024). Бързо ръководство WordPress. София. [АлексСофт](https://www.orangecenter.bg/knizharnitsa/nehudozhestvena-literatura/informatsionni-tehnologii?publisher=%D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%A1%D0%BE%D1%84%D1%82)

Freeman, А. (2024). Pro ASP.NET Core 6. Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. София. APress

Коцев, Н., Коцев, Е. & Рускова, С. (2013). Основи на управлението. Русе: Авангард принт

MongoDB Inc. (2024). How to use Mern stack: a complete guide. Достъпно към 25.11.2024г. от <https://www.mongodb.com/resources/languages/mern-stack-tutorial>

Microsoft Learn Challenge. Първи стъпки с ASP.NET MVC. Достъпно към 25.11.2024г. от <https://learn.microsoft.com/bg-bg/aspnet/mvc/overview/getting-started/introduction/>

# Приложение 1

Съдържание на източници. …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

# Приложение 2

Съдържание на източници. …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

# Приложение 3

Съдържание на източници. …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

# Приложение 4

Съдържание на източници. …………………… ………… ……… ………….. ……………… ………………….. ……………………………. …………… ………………… ………

1. Next.js, Inc., 2025, Next.js Documentation, достъпен на 03.01.2025 от https://nextjs.org/docs. [↑](#footnote-ref-1)
2. Microsoft Corporation, 2025, TypeScript Documentation, достъпен на 03.01.2025 от https://www.typescriptlang.org/docs/. [↑](#footnote-ref-2)
3. Firebase, Inc., 2025, Firebase Documentation, достъпен на 03.01.2025 от https://firebase.google.com/docs. [↑](#footnote-ref-3)
4. Amazon Web Services, Inc., 2025, Amazon S3 Documentation, достъпен на 03.01.2025 от <https://docs.aws.amazon.com/s3/>. [↑](#footnote-ref-4)
5. Meta Platforms, Inc., 2025, React Documentation, достъпен на 03.01.2025 от <https://react.dev/>. [↑](#footnote-ref-5)
6. Mozilla Developer Network, 2025, CSS Documentation, достъпен на 03.01.2025 от <https://developer.mozilla.org/en-US/docs/Web/CSS>. [↑](#footnote-ref-6)
7. SuperHosting, 2025, Какво е HTML(HyperText Markup Language), достъпен на 03.01.2025 от https://help.superhosting.bg/html-hypertext-markup-language.html [↑](#footnote-ref-7)
8. Medium, 2025, Nextjs is Backend or frontend, достъпен на 12.02.2025 от <https://medium.com/@turingvang/nextjs-is-backend-or-frontend-d7b6da5f2597> [↑](#footnote-ref-8)
9. FireBase, 2025, Cloud FireStore Data Model, достъпен на 13.02.2025 от <https://firebase.google.com/docs/firestore/data-model> [↑](#footnote-ref-9)