|  |  |  |
| --- | --- | --- |
| **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: УЕБ ПРИЛОЖЕНИЕ ЗА РАЗГОВОРИ И ОБМЯНА НА СЪОБЩЕНИя**

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

/Симеон Бухалов/ /Катя М. Семкова/

Клас: 12а

e-mail: sbuhalov3@gmail.com0

Пловдив

2024 година

# СЪДЪРЖАНИЕ

[СЪДЪРЖАНИЕ 0](#_Toc163579017)

[УВОД 1](#_Toc163579018)

[ГЛАВА 1 – ТЕОРЕТИЧНИ ОСНОВИ И КОНЦЕПЦИИ ПО ТЕМАТА 3](#_Toc163579019)

[1.1. Обзор на технологични алтернативи 3](#_Toc163579020)

[1.1.1. LAMP стек 3](#_Toc163579021)

[1.1.2. MERN стек 4](#_Toc163579022)

[1.1.3. SolidJS стек 5](#_Toc163579023)

[1.1.4. Django стек 6](#_Toc163579024)

[1.1.5. Svelte стек 6](#_Toc163579025)

[1.1.6. ASP.NET стек 7](#_Toc163579026)

[1.2. Избор на подходящи технологии за текущата разработка 8](#_Toc163579027)

[1.3. Специфики на ASP.NET – Svelte стека 9](#_Toc163579028)

[1.3.1. Svelte 9](#_Toc163579029)

[1.3.2. SvelteKit 10](#_Toc163579030)

[1.3.3. ASP.NET Core като Backend 11](#_Toc163579031)

[1.3.4. EntityFrameworkCore и MSSQL 11](#_Toc163579032)

[1.3.5. ASP.NET Core Identity 12](#_Toc163579033)

[1.3.6. Node.js 12](#_Toc163579034)

[1.3.7. Node Package Manager 13](#_Toc163579035)

[1.3.8. NuGet Package Manager 13](#_Toc163579036)

[1.3.9. Други технологии 14](#_Toc163579037)

# УВОД

В днешната цифрова ера, комуникацията е сърцето на всяко социално и бизнес общество. От момента, в който Интернетът стана неизменна част от ежедневния живот, потребителите се нуждаят от инструменти, които да им осигурят бърза, ефективна и сигурна комуникация с техните колеги, приятели и близки.

Онлайн общуването позволява на потребителите да създават много повече връзки с други хора от различни места по-света, което би подпомогнало за развитието на човека. То преодолява ограниченията на далечните разстояния и несъответствия в дневния график на общуващите, като им предоставя способността до комуникират от където и да е стига да имат нужното устройство.

Целта на разработката е създаването на чат приложение, което предлага ползотворни услуги за потребителите си. С помощта на приложението, те могат да създават групи, където да споделят идеи, информация и преживявания с колеги, семейство или приятели. В същото време, потребителите могат да добавят приятели и да комуникират с тях по индивидуален начин, без да са ограничени от присъствието на нежелани потребители.

За да бъде успешно реализирането на приложението, е необходимо да се изследват методите и технологиите, които ще бъдат използвани в процеса на разработка. Трябва да се представи как тези методи и технологии ще бъдат приложени и как ще се развиват функционалностите им. Това включва създаването на интерактивен уебсайт, който дава възможност на потребителите да създават акаунти и да комуникират един с друг.

В първа глава ……… ……………. ……………… ……….. …. …………. …… ………………………… ……………………….

Във втора глава ……… ……………. ……………… ……….. …. …………. …… ………………………… ……………………….

В трета глава ……… ……………. ……………… ……….. …. …………. …… ………………………… ……………………….

В четвърта глава ……… ……………. ……………… ……….. …. …………. …… ………………………… ……………………….

В пета глава ……… ……………. ……………… ……….. …. …………. …… ………………………… ……………………….

# ГЛАВА 1 – ТЕОРЕТИЧНИ ОСНОВИ И КОНЦЕПЦИИ ПО ТЕМАТА

## Обзор на технологични алтернативи

При планирането на разработката са разучени подробно възможните софтуерни технологии и методи за самообучение с целта да се намери най-ефективното и потребителски-интуитивно решение. Технологиите, които улавят вниманието в случая са: LAMP, MERN, SolidJS, Django, Svelte и ASP.NET стекове. Подходящият избор на стек и технологии има ключово значение не само за разработката на проекта, а и за потребителя. Правилните технологии, когато се подберат спрямо изискванията на задачата, значително улесняват осъществяването ѝ, а същевременно подсигуряват потребителското взаимодействие с продукта, защитавайки го от технически проблеми или от проблеми в производителността на самия продукт. Изчерпателния анализ на разгледаните технологии цели да подсигури адаптацията и връзката между стек-а и концепцията, която трябва да бъде реализирана. В по-нататъчни етапи, избора и анализа позволяват и лесната поддръжка на продукта от разработчиците.

### RoR стек

**Ruby on Rails (RoR)** е популярен Full Stack фреймуърк, който използва Ruby за сървърната част и обработва всичко - от базата данни до интерфейса на потребителя. RoR е известен със своята простота, елегантност и способността си да ускори процеса на разработка.

**Ruby** е език за програмиране, който е създаден с акцент върху четимостта на кода. Ruby е динамичен, обектно-ориентиран език, който улеснява разработката.

**Ruby on Rails (RoR)** е уеб фреймуърк, който се базира на Ruby и предоставя множество вградени функции, които ускоряват процеса на разработка. Той следва принципите на Convention over Configuration и Don't Repeat Yourself (DRY), което улеснява разработчиците да се концентрират върху бизнес логиката на приложението, а не на конфигурацията и първоначалната настройка. RoR поддържа различни релационни бази от данни като PostgreSQL, MySQL и SQLite.

Предимства на този стек са, че има голяма колекция от компоненти, които са готови за ползване. Те могат да бъдат достъпени от пакетния мениджър на Ruby – RubyGems. Благодарение на тези готови компоненти, разработката става много по-бърза, поради по-малкото писане на код. Друго предимство е, че RoR има вградена защита срещу различни видове атаки, като някои от тях са SQL инжекции и CSRF.

Недостатъци могат да бъдат времето за научаване на Ruby, както и липсата на ресурси въпреки активната общност на RoR.

### MERN стек

MERN се състои от 4 технологии: MongoDB, Express.js, React и

Node.js. Известен е с това, че е гъвкав, което го прави удобен за работа без значение от изискванията на проекта.

**MongoDB** е NoSQL база данни, която предлага гъвкава и мащабируема структура. Тя използва JSON-подобни документи и е идеална за уеб приложения, които изискват скорост при обработка на данни.

**Express.js** е минималистичен уеб фреймуърк за Node.js, който предоставя удобен начин за създаване на уеб сървъри и управление на маршрутирането. Той също така улеснява работата с бази данни и взаимодействието с други уеб услуги.

**React** е библиотека за създаване на потребителски интерфейси, която предоставя голям набор от инструменти за създаване на динамични и интерактивни уеб приложения. Тя използва компонентен подход и виртуален DOM за оптимизация на производителността.

**Node.js** е сървърна платформа, която позволява на разработчиците да създават уеб приложения използвайки JavaScript както на клиентската, така и на сървърната страна. Тя е изключително гъвкава и мащабируема, което я прави идеална за разработка на съвременни уеб приложения.

MERN стека е много добър поради причината, че всички технологии, които се използват в него са на JavaScript, което улеснява връзката между фронтенда и бекенда.

Недостатъци на този стек са сложността му за неопитни разработчици. Поради това, че MongoDB е NoSQL база данни, разработчиците се нуждаят от повече опит, за да може да е напълно ефикасна базата данни. Друг такъв недостатък е това, че React е базиран на компоненти и библиотеки, което го прави много зависим от поддръжката на тези компоненти. Проблемът идва там, че библиотеките не са създадени от създателите на React и не винаги са добре оптимизирани.

### SpringBoot стек

**Spring Boot** е Full Stack фреймуърк за Java, който се използва за създаване на съвременни уеб приложения и микросервизи. Той е базиран на Spring Framework и предоставя бърз и лесен начин за стартиране на нови проекти с минимална конфигурация. Основни части от стека са: Spring Framework, Spring Boot, Spring Data, Spring Security, Spring Web MVC

**Spring Framework** е основният компонент на стека, който предоставя инфраструктурна поддръжка за разработка на Java приложения.

**Spring Boot** е слой над Spring Framework, който улеснява стартирането и конфигурацията на приложения. Spring Boot автоматизира много от конфигурационните задачи и така подпомага за бързо разработване на приложения.

**Spring Data** предоставя абстракция над различни видове бази данни и предоставя лесен начин за извършване на операции като CRUD (Create, Read, Update, Delete). Spring Data улеснява взаимодействието с базите данни и намалява необходимостта от писане на ръчен SQL код.

**Spring Security** отговаря за автентикацията и авторизацията в приложенията. Предоставя вградени начини за управление на потребителски роли, сесии, токени и други сигурностни аспекти.

**Spring Web MVC** предоставя възможност за създаване на уеб приложения със Spring. Spring Web MVC е традиционен модел-изглед-контролер (MVC) фреймуърк.

Предимства на Spring Boot са обширният набор от инструменти за разработка (Spring Cloud, Spring Data, Spring Security и др.) както и това, че автоматизираната конфигурация ускорява процеса на разработката.

Един от недостатъците е, че въпреки автоматизираната конфигурация, на разработчиците все пак може да им е трудно да научат самият стек, най-вече ако до сега не са работили с Java. Друг проблем е изискването на ресурси на стека, поради големият набор от инструменти – трябва да се подбере правилен хостинг, който може да издържи на по-голямо натоварване.

### Django стек

**Django** е Full Stack уеб фреймуърк, базиран на Python, който улеснява и ускорява разработката на уеб приложения със своята гъвкавост и готовите за употреба компоненти. Основни компоненти за стека са: Django Framework, Django ORM, Django Templates, Django Forms, Django Admin.

**Django Framework**: Основният компонент на стека, който предоставя цялостно решение за уеб разработката. Django предлага вградени функционалности като ORM (Object-Relational Mapping), маршрутиране, форми, административен интерфейс и много други.

**Django ORM**: Обектно-релационната система на Django улеснява взаимодействието с бази данни чрез Python код, като автоматизира много от задачите, свързани с манипулирането на данни.

**Django Templates**: Предлагат лесен и гъвкав начин за генериране на динамичен HTML код. Шаблоните на Django позволяват лесно разделяне на логиката и представянето на уеб приложенията.

**Django Forms**: Улесняват създаването и валидацията на формуляри в уеб приложенията. Django Forms автоматизират процеса на обработка на формулярни данни и управление на взаимодействието с потребителите.

**Django Admin**: Вграденият административен интерфейс на Django предоставя готово за употреба решение за управление на данни, потребители и права в уеб приложенията.

Предимства са бързата разработка благодарение на компонентите, които предлага, безопасността – има вградена защита от CSRF и SQL инжекции и има активна общност, което предоставя различни библиотеки за ползване в проектите, както и ресурси за учене на стека.

Един голям недостатък е, че Django улеснява много от задачите на разработчиците, но въпреки това, той може да бъде по-сложен за научаване, особено за начинаещи в Python и уеб разработката.

### Svelte стек

**Svelte** е модерен JavaScript фреймуърк за създаване на интерактивни уеб приложения с прост и елегантен подход към разработката. Той предлага различни инструменти, улесняващи разработката като: Svelte Components, Svelte Stores, Svelte Routing и др. **SvelteKit** е официалният маркетингов фреймуърк на Svelte, който предоставя интегрирано и мощно решение за разработката на уеб приложения със Svelte. Вместо традиционния подход с виртуален DOM, Svelte прилага компилация на кода, генерирайки чист JavaScript, който работи директно с DOM-a. Тази архитектура е разработена с идеята да улесни процеса на създаване на уеб приложения, като премахва необходимостта от излишен повтарящ се код.

В основата на Svelte лежи чистият JavaScript, съчетан с функционалности, които автоматично въвеждат реактивност в кода. Например, променливите стават реактивни по подразбиране – актуализират DOM-a динамично в момента в който се променят.

**SvelteKit** допълва Svelte, предоставяйки интегрирана среда за разработка и богат набор от инструменти. Той е проектиран да улесни работата на разработчиците, като предлага автоматична маршрутизация, интеграция с Vite и опции за статично и динамично рендериране на уеб страниците.

**Svelte Components**: В Svelte, всичко е компонент. Компонентите са изградени от HTML, CSS и JavaScript код, които са организирани в еднофайлови компоненти (.svelte файлове). Това улеснява преизползването, тестването и поддръжката на кода.

Svelte Store играе централна роля в управлението на състоянието в Svelte стека то е прост и ефективен начин за управление на състоянието на приложението и комуникация между компонентите.

Предимства са прозиводителността, заради начина, по който се компилира кода, чистотата, заради компонентите, предлагани от Svelte, които предотвратяват повтарящият се код и гъвкавостта – поради чистият JavaScript код е много лесно интегрирането на библиотеки и инструменти.

Svelte и SvelteKit са сравнително нови технологии, затова един недостатък е, че хората, ползващи ги не са много и няма много официална документация.

### ASP.NET стек

**ASP.NET MVC** е популярен и мощен технологичен стек за разработка на уеб приложения, базиран на .NET платформата. Той предлага структуриран и модулен подход към създаването на уеб приложения, който разделя логиката за представяне, бизнес логиката и данните в различни компоненти на приложението. Основните му части са: Razor View Engine, Entity Framework, ASP.NET Identity и ASP.NET Web API, като най-предпочитаната база данни за ползване с него е SQL Server поради лесната комуникация с Entity Framework.

**Razor View Engine**: Разширение на ASP.NET MVC, което позволява създаването на динамични HTML страници, комбинирайки HTML и C# код в един и същи файл.

**Entity Framework (EF)**: ORM (Object-Relational Mapping) инструмент, който улеснява взаимодействието с бази данни чрез предоставяне на удобни API за операции с данни.

**ASP.NET Identity**: Библиотека за управление на идентичността, която предоставя функционалности като управление на потребители, автентикация и авторизация.

**ASP.NET Web API**: Фреймуърк за създаване на HTTP базирани услуги, което улеснява интеграцията на уеб приложения с различни клиентски платформи.

ASP.NET стекът е много добър с това, че благодарение на ASP.NET Identity сигурността се имплементира много по-лесно, има голяма общност, следователно има много ресурси, документацията е лесна за четене и интеграцията на други .NET инструменти и технологии е по-лесна.

Недостатък на този стек е зависимостта му от Windows – употребата му и работата с него е силно ограничена на различни операционни системи различни от Windows

## Избор на подходящи технологии за текущата разработка

При разработването на чат приложение, е важно да се изберат подходящите технологии, които да осигурят ефективна, скалируема и сигурна реализация на проекта. **Svelte** и **SvelteKit** представляват модерни и иновативни технологии за уеб разработка, които са идеални за създаване на динамични и интерактивни уеб приложения като чат системи. С тях управлението на състоянието на страниците става лесно и бързо, което е ключово за обмяната на съобщения между потребители.

**ASP.NET MVC** е утвърден и надежден бакенд стек, който предлага широк набор от функционалности и инструменти за разработване на уеб приложения. Той е добре поддържан, което гарантира сигурността на бакенда, както и данните на потребителите. Заради възможността за интеграция на SQL Server с Entity Framework, комуникацията с базата данни става много лесна, което осигурява ефективността на управлението на данните. Благодарение на механизмите за автентикация и авторизация на ASP.NET Identity потребителите могат та очакват високи стандарти за сигурност, което е критично за защитата на данните им в приложението.

## Специфики на ASP.NET – Svelte стека

Комбинирането на Svelte с ASP.NET създава архитектура за уеб приложения, комбинираща иновативен и бърз фронтенд с мощен и гъвкав бакенд.

### Svelte и SvelteKit за клиентска част

Svelte преобразува декларативния код в оптимизиран JavaScript преди зареждането на приложението, като осигурява по-бързо и плавно потребителско изживяване. Той се отличава с ясен и директен синтаксис, улесняващ разработката и поддържането на сложни интерфейси и предоставя компоненти за разработчиците. Чрез компонентите, програмистите могат да избегнат повтарянето на един и същ код, като ползват един компонент с различни параметри вместо да пренаписват кода всеки път. Това прави кодът по-чист и по-лесен за надграждане и работене върху него.

SvelteKit предоставя гъвкава разработческа среда и структура, ускоряващи началото на нови проекти. Интегрираните решения за маршрутизация и състояние улесняват създаването на уеб приложения с оптимизирано кеширане.



Фигура 1 – Лого/Банер на Svelte

### ASP.NET Core за сървърна част

ASP.NET Core осигурява мощна и безопасна платформа за разработка на уеб API. Разполага с много добра защита срещу различни видове атаки като SQL инжекции, като така данните на потребителите биват предпазени. Разработката на Web API става чрез задаване на Endpoint и създаване на функционалност за този Endpoint. Със своите инструменти и библиотеки, тя улеснява интеграцията и поддръжката на сигурни бакенд решения.

### EntityFrameworkCore и MSSQL

Entity Framework Core (EFC) представлява интегрирано средство за управление на бази данни, което улеснява връзкат между приложенията и базите данни във фреймуърка на ASP.NET Core. Този ORM инструмент преобразува сложните операции с база данни в прости програмни операции, което опростява CRUD операциите.

Съвместно с SQL Server Management Studio (SSMS) и Microsoft SQL Server (MSSQL), Entity Framework Core предлага мощна платформа за управление на релационни данни. MSSQL е високопроизводителна и надеждна система за управление на бази данни, която предлага богат набор от функционалности и възможности за управление на данните. Тази комбинация осигурява възможност за създаване на устойчиви и мащабируеми уеб приложения, които могат да се адаптират към различни бизнес изисквания и натоварване.

### ASP.NET Core Identity

ASP.NET Core Identity е интегриран модул в ASP.NET Core, предоставящ гъвкави и мощни инструменти за управление на потребителски идентичности в уеб приложенията. Този модул улеснява регистрацията, автентикацията, авторизацията и управлението на потребителски профили, като предоставя високо ниво на сигурност и защита на потребителските данни.

Модулът предлага възможности за двуфакторна аутентикация, управление на пароли, потребителски роли и права, както и възможност за връзка с външни идентичности като Google, Facebook и други.

### Node.js и Node Package Manager

Node.js е мощна и гъвкава платформа за създаване на сървърни приложения, базирана на JavaScript. Тя пoмага на разработчиците да създадат ефикасно уеб приложение, което може да обработва заявки паралелно, без забавяне.

Една от ключовите характеристики на Node.js е неблокиращата I/O операция, която позволява оптимизиране на изпълнението на заявките и увеличаване на скоростта на приложението. Това прави Node.js идеален избор за реалновременни приложения, като чат системи, стрийминг платформи и социални мрежи.

С Node.js, разработчиците имат достъп до голям брой модули и библиотеки чрез npm (Node Package Manager), което улеснява интеграцията с различни технологии и инструменти. Това поддържа ефективната разработка и ускорява процеса на създаване на функционалности и модули в сървърната част на приложението.



Фигура 2 – Лого на Node.js

## NuGet Package Manager

NuGet е инструмент, който подпомага процеса на добавяне, интегриране и поддръжка на разнообразни пакети и библиотеки в .NET приложения. Чрез интуитивен интерфейс и функционалност, NuGet дава възможност на разработчиците да избират и интегрират ключови компоненти и решения в техните проекти, осигурявайки така ускорен и оптимизиран процес на разработка. Той предоставя инструменти за лесно търсене, инсталация и актуализация на пакети, като така допринася за непрекъснатото развитие и поддържане на висококачествените .NET приложения.

### Vercel за Frontend хостинг

Vercel е платформа за разработка и хостинг, специализирана в оптимизацията и разгръщането на фронтенд приложения. Създадена с фокус върху производителността и удобството за разработчиците, Vercel предоставя едностепенно решение за създаване, тестване и публикуване на уеб приложения, което обединява всички етапи на разработката в една интуитивна и ефективна работна среда.

Vercel се интегрира напълно с популярни системи за управление на версии като GitHub, GitLab и Bitbucket, позволявайки автоматизирана актуализация при всяка промяна на кода.

### MyWindowsHosting за Backend хостинг

MyWindowsHosting е водеща платформа за уеб хостинг, специализирана в предоставянето на надеждни и мащабируеми услуги за хостинг на бакенд приложения, базирани на различни технологии и платформи. Със своя фокус върху Windows базирани решения, MyWindowsHosting предлага обширен набор от функции и инструменти, които улесняват процеса на разработка, разгръщане и управление на бакенд инфраструктурата. Платформата също така предлага уникалната възможност за 2 месеца безплатен пробен период, позволявайки на потребителите да изпробват и оценят качествата и предимствата ѝ преди да се ангажират с дългосрочен план.

MyWindowsHosting предоставя интегрирана поддръжка за популярни Microsoft технологии като ASP.NET, .NET Core, SQL Server и други, което осигурява съвместимост и оптимизирана производителност на бакенд приложенията.

Платформата поддържа WebDeploy, което позволява лесно и бързо качване на уеб приложения и обновления безпосредствено от разработческата среда, като гарантира синхронизирането на кода и конфигурациите между разработческата машина и хостинг сървъра.

MyWindowsHosting предлага цялоденонощна техническа поддръжка и екип от професионалисти, готови да помогнат при всякакви проблеми или въпроси, свързани с хостинга и инфраструктурата.

### SignalR за комуникация в реално време

SignalR е библиотека за ASP.NET Core, която предоставя прост и мощен начин за реализация на реално-временна комуникация между уеб клиенти и сървъри. Този инструмент улеснява създаването на интерактивни и динамични уеб приложения, които могат да обменят данни в реално време, без необходимостта от постоянни заявки към сървъра.

SignalR позволява моменталното изпращане на съобщения и обновления от сървъра към уеб клиентите, като гарантира актуалност и динамичност на предоставяната информация.

### Други технологии

CORS (Cross-Origin Resource Sharing) – стандартна техника за контролиране на достъпа до ресурсите на уеб приложения, позволявайки ограничен достъп до източници на различни домейни. Този механизъм е особено полезен при разработването на уеб приложения с архитектура, където ресурсите се предоставят от различни домейни или услуги.

JasonWebToken – популярен механизъм за автентикация в ASP.NET приложенията, използващ JSON Web Tokens (JWT) за предоставяне на сигурни и компактни токени за удостоверяване на потребителите. Този подход осигурява сигурен и ефективен начин за проверка на идентичността на потребителите и управление на достъпа до ресурсите в уеб приложенията.

Bootstrap – edin от най-популярните и използвани фронтенд фреймуърци в света на уеб разработката. Този мощен инструмент предоставя обширен набор от предварително дефинирани стилове, компоненти и скриптове, които улесняват и ускоряват процеса на създаване на съвременни и атрактивни уеб приложения.

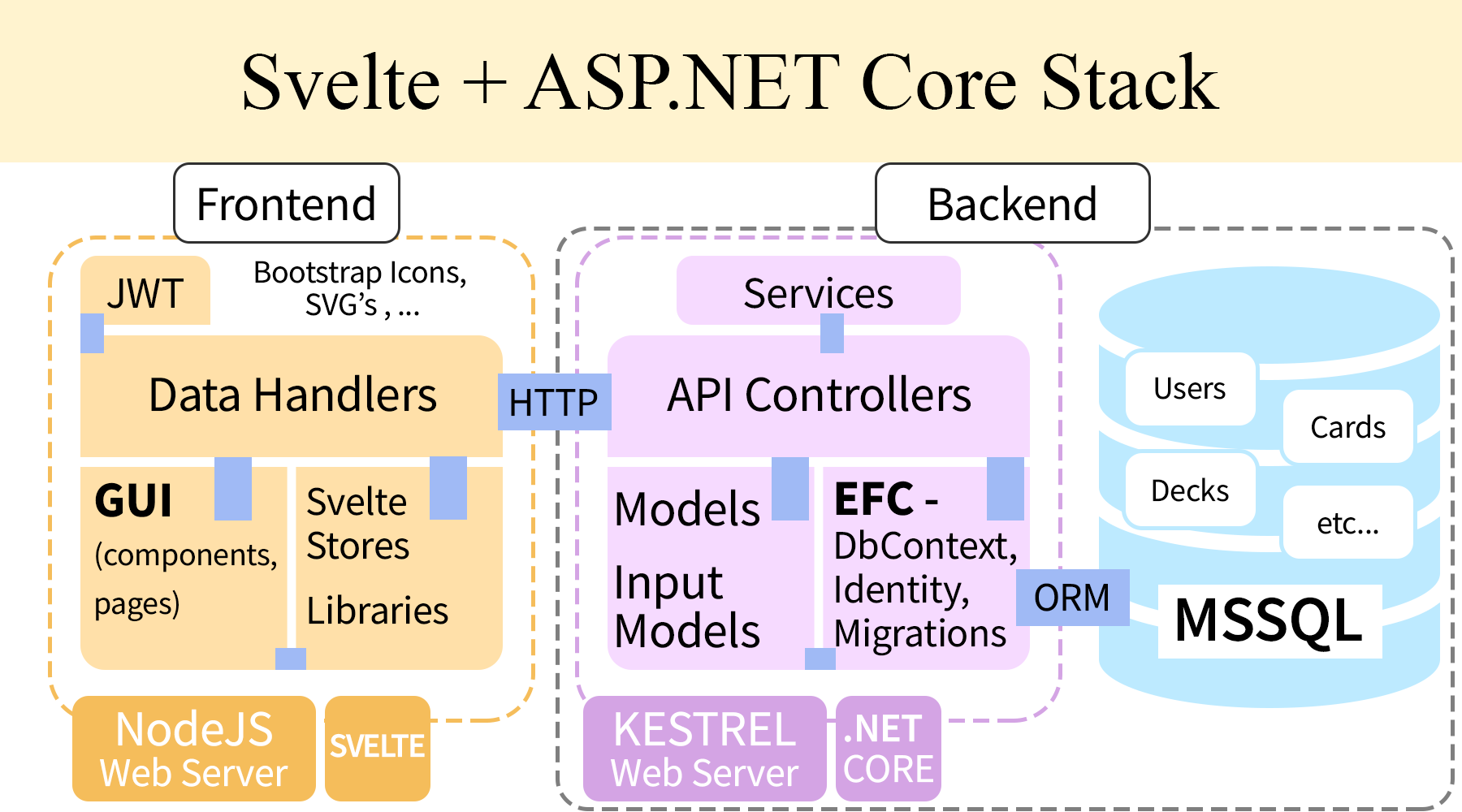
FontAwesome – набор от икони, предоставящ широка гама от векторни изображения, които могат да бъдат лесно интегрирани в уеб приложенията. Тези икони добавят допълнителен естетически елемент и функционалност към интерфейса, без да се налага да се използват сложни графични редактори или специализирани изображения.

Cropper.js – мощна и гъвкава JavaScript библиотека, която предоставя функционалност за обработване и редактиране на изображенията директно в уеб браузъра. Тази библиотека е изключително полезна за разработчиците, които работят с уеб приложения, които изискват манипулиране на изображения, като профилни снимки, галерии или редактори за изображения.

# ГЛАВА 2 – СТРУКТУРА НА РАЗРАБОТКАТА

## 2.1. Обща схема

Проектът се разработва с цел създаване на съвременно и функционално уеб приложение, което комбинира най-добрите практики и технологии от фронтенд и бакенд разработката. Разделя се на две части – Frontend и Backend, които са хостнати с различни услуги. Frontend-а е хостнат на Node.js сървър чрез Vercel, а Backend-a е хостнат на KESTREL сървър чрез MyWindowsHosting, като базата данни е част от backend-a.



Фигура 3 – Схема на структурата на проекта

### 2.1.1. Клиентска част (Frontend)

Фронтендът на проекта е базиран на модерния Svelte фреймуърк, с поддръжка от SvelteKit за улеснено разработване и мащабируемост. Основната му задача е да предоставя динамичен и интерактивен потребителски интерфейс, използвайки силата на Svelte's компилатор за генериране на ефективен JavaScript код.

Фронтендът е организиран чрез TypeScript Data Handlers, които са отговорни за обработката на HTTP заявките към бекенда. Тези хендлъри обработват информация в JSON формат и предоставят необходимите обектни интерфейси за конверсия на данни в TypeScript обекти. За сигурността на заявките, JWT токенът се добавя към всяка заявка след успешна аутентикация на потребителя. Обектите могат да бъдат пренесени към бекенд-а направо през URL-a на заявката като параметри или като се запишат като част към тялото ѝ.

Допълнително, Svelte хранилищата (stores) се използват за управление на реактивните данни между различните компоненти на приложението. Тези хранилища позволяват лесно обменяне на данни и обновяване на интерфейса при промяна на състоянието. Примери за използване на такива хранилища в проекта са: Информацията за сегашния потребител; Заявките за приятелство; Приятелите на текущият потребител и др.

За обмяната на съобщения в реално време отговарят Hubs (хъбовете) във фронтенда. Те създават връзка към хъб с определен еndpoint в бакенда. Всеки потребител е свързан с уникално ID на връзката си към хъба. След създаване на връзката биват дефинирани Event Listeners за различни действия, които могат да се случат, и при получаване на такъв Event от бакенда, функцията за съответния listener се изпълнява. Това предоставя двустранна комуникация между фронтенда и бакенда, като позволява бързо и ефективно обменяне на данни и съобщения без да има нужда от презареждане на страницата.

За визуалната част на приложението са включени векторни графики и икони от FontAwesome, а за стилизацията се използва частично Bootstrap и обикновен CSS.

### 2.1.2. Сървърна част (Backend)

Backend-ът на проекта е разработен с използване на ASP.NET Core Framework. Централната му роля е да предоставя API за взаимодействие с клиентската част на приложението. Вместо традиционните Views, Backend-ът работи като API, като контролерите са от типа ApiController, което позволява гъвкаво управление на URL пътеките за HTTP заявките от Frontend-а.

За създаване на сигурни и компактни токени за удостоверяване на потребителите, в проекта е интегриран JWTService. Този сервиз е специализиран за генериране на JWT токени при успешен вход или регистрация на потребителите. При заявки, изпратени от фронтенда бива прикачен този токен, и ако endpoint-а не получи токена, или токена е невалиден, то потребителят не бива авторизиран.

Backend-ът на разработката използва EntityFrameworkCore – ORM библиотеката на ASP.NET Core и MSSQL. Тази библиотека позволява създаването, модифицирането и изтриването на таблици или колони въз основа на дефинирани модели и DbContext-а на приложението. За управление на промените в схемата на базата от данни, EntityFrameworkCore използва миграции. Миграцията включва необходимия код за отразяване на новите промени в MSSQL, както и еквивалентния код за премахване на тези промени.

Моделите в проекта дефинират структурата на данните, с които работи приложението. Чрез EntityFrameworkCore, разработчиците могат да управляват тези модели и да синхронизират промените с базата данни. Освен основните модели, са създадени и Input модели. Те служат за конвертиране на JSON данни от Frontend-а в структурирани обекти, готови за обработка от Backend-а. ViewModel-ите са още една част от архитектурата, които предоставят специфични представления на моделите, оптимизирани за нуждите на различните изгледи и операции в приложението.

### 2.2. Структура на Базата от данни

Проектът използва релационна SQL база данни с MSSQL като система за управление. Базата данни е организирана в различни таблици, свързани чрез полета за вторичен ключ, които сочат към първичните ключове на други таблици.

Базата съдържа различни релации (таблици), които могат да бъдат свързани помежду си. Тази връзка се извършва посредством първичните ключове на редовете в таблиците. Връзките от тип many-to-many (един ред в таблицата може да е свързан с много от друга или обратното) се извършват чрез създаване на отделна таблица за връзка. Тази трета таблица не се нуждае от първичен ключ, а съдържа само две полета – външни ключове към обектите от другите таблици. Пример за такава „посредническа“ таблица може да е UserRoles таблицата, която съдържа външни ключове към потребител и роля.

Някои от таблиците са ръчно създадени, а другите автоматично от ASP.NET Identity. Ръчно-създадените таблици са генерирани чрез модели и миграции с помощта на Entity Framework Core (EF Core). EF Core позволява разработчиците да дефинират модели за техните данни, които автоматично се превръщат в SQL таблици.

### EFC и Identity таблици

Таблиците от ASP.NET Identity управляват потребителската идентичност и авторизация в приложението. Те включват **AspNetUsers**, **AspNetRoles**, **AspNetUserRoles**, **AspNetUserClaims**, **AspNetUserLogins** и **AspNetUserTokens**. Въпреки това, че AspNetUsers е генерирана автоматично, тя все пак наследява допълнителни полета от ApplicationUser модела, като те са: ProfilePictureUrl; AccountCreatedDate; Status, Bio и IsAdmin.