# Списък с цитирани фигури

[Фигура 4.1.2.1.: Дефиниция на таблица за потребителски профили](#_Toc194096553)

[Фигура 4.1.2.2: Дефиниция на таблица за предлагане на приятелства](#_Toc194096554)

[Фигура 4.1.2.3.: Дефиниция на таблица за чат стаи](#_Toc194096555)

[Фигура 4.1.2.4:Дефиниция на таблица за съобщения](#_Toc194096556)

[Фигура 4.1.2.5.:Диаграма на базата данни](#_Toc194096557)

[Фигура 4.2.1.1.:Избор при създаване на уеб приложение](#_Toc194096558)

# Увод

В съвременният свят комуникацията в реално време е от голямо значение за

разбирателството между хората. С постоянното развитие на технологии нараства необходимостта от гладка, бърза и надеждна комуникация, чат системите играя важна роля както в личния, така и в професионалния живот.

В дипломният проект, ще представя своята разработена чат система, като използвам технологиите Next.js, Supabase и Tailwind CSS. Основната цел на проекта е създаването на функционална и удобна платформа за изпращане и получаване на съобщения между потребителите, както и управление на приятелства и покани.

Очакваните резултати включват изграждане на стабилна и сигурна система, която осигурява гладка комуникация, оптимизирано потребителско изживяване и ефективно управление на потребителските данни.

# Глава първа: Теоретична част

## Цел на проекта

В съвременния дигитален свят комуникацията в реално време е неизменна част от ежедневието на хората. Съществуват множество платформи като Discord, Viber, Messenger и други, които осигуряват възможност за бърза и ефективна връзка между потребителите. Настоящият дипломен проект има за цел разработването на уеб-базирана чат система, която позволява обмен на съобщения в реално време, както и управление на списък с контакти чрез функционалност за изпращане и приемане на приятелски покани.

За постигането на тази цел е необходимо реализирането на следните ключови задачи: Създаване на уеб приложение чрез Next.js

* + 1. Създаване на уеб приложение с Next.js

Проектът ще бъде разработен с помощта на Next.js – популярен framework за React, който осигурява висока производителност, оптимизация и ефективно изграждане на динамични уеб приложения. Използването му ще допринесе за подобряване на цялостното потребителско изживяване и бързодействието на системата.

* + 1. Изграждане на база данни със Supabase

Данните, свързани с потребителите, изпратените съобщения и приятелските връзки, ще бъдат съхранявани в база данни, изградена със Supabase – облачна платформа, предоставяща back-end услуги и база данни в реално време. Supabase предлага също така автентикация и управление на потребителски достъп, което допринася за сигурността на системата.

* + 1. Реализиране на комуникация в реално време

Основна функционалност на приложението е изпращането и получаването на съобщения в реално време, без необходимост от презареждане на страницата. Това ще бъде постигнато чрез Supabase Realtime, който осигурява автоматична синхронизация на данните между участниците в разговора.

* + 1. Разработване на система за приятелства

За да могат потребителите да комуникират, ще бъде изградена система за изпращане, приемане и отказване на приятелски покани. Тази информация ще бъде организирана в отделни таблици в базата данни, като ще се осигури интуитивен интерфейс за управление на контактите.

* + 1. Имплементиране на система за регистрация и автентикация

Достъпът до платформата ще бъде ограничен до регистрирани потребители. Регистрацията и входът в системата ще се осъществяват чрез Supabase Auth, който ще предоставя сигурен механизъм за удостоверяване на потребителите чрез имейл и парола.

* + 1. Разработване на интуитивен потребителски интерфейс

Визията и удобството на потребителския интерфейс са от съществено значение за успешното използване на системата. За тази цел ще бъде приложен Tailwind CSS, който позволява изграждане на модерен, адаптивен и интуитивен дизайн. Визията и удобството на потребителския интерфейс са от съществено значение за успешното използване на системата. За тази цел ще бъде приложен Tailwind CSS, който позволява изграждане на модерен, адаптивен и интуитивен дизайн.

* + 1. Хостване на проекта и осигуряване на достъпност

За да бъде достъпно за потребителите, уеб приложението ще бъде хостнато чрез Vercel – платформа, специализирана за разгръщане на Next.js проекти. Това ще осигури лесен достъп до системата, както и непрекъсната наличност на услугите. За да бъде достъпно за потребителите, уеб приложението ще бъде хостнато чрез Vercel – платформа, специализирана за разгръщане на Next.js проекти. Това ще осигури лесен достъп до системата, както и непрекъсната наличност на услугите.

* + 1. Очаквани резултати

В резултат от изпълнението на горепосочените задачи се очаква създаването на функционална, надеждна и сигурна уеб-базирана чат система, която осигурява ефективна комуникация в реално време. Проектът ще демонстрира възможностите на Next.js и Supabase за изграждане на модерни уеб приложения, като същевременно гарантира добро потребителско изживяване и висока производителност.

## Съществуващи решения

В съвременния дигитален свят съществуват множество чат платформи, които предоставят възможност за комуникация между потребители. Сред най-популярните са Messenger, Viber, Discord и други. Тези системи предлагат разнообразни функции, включително текстова и гласова комуникация, групови чатове, мултимедийни съобщения и интеграция с различни услуги.

Въпреки широката им популярност, съществуващите решения често са обвързани със специфични екосистеми, изискват регистрация с телефонен номер или социален профил и понякога налагат ограничения върху начина, по който потребителите могат да взаимодействат.

Създаването на нова чат система дава възможност за персонализиран подход и внедряване на специфични функции според нуждите на проекта. Основният фокус на настоящото решение е опростената комуникация в реално време, лесното управление на приятелства и съобщения, както и интеграцията на модерни технологии като Next.js, Supabase и Tailwind CSS за оптимална работа и удобен интерфейс.

# Глава втора: Практическа част

## Проектиране на системата

### Архитектура на приложението

Проектът е разделен на три основни части:

Frontend частта от приложението е тази, с която потребителят взаимодейства. За нейното изграждане ще се използват Next.js и Tailwind CSS. Next.js ще бъде използван за създаване на интерфейса, навигацията и динамичното зареждане на съдържание, докато Tailwind CSS ще се използва за дизайна на самия интерфейс, което ще позволи бързо създаване и адаптиране на компоненти към различни устройства. Компонентите на frontend частта включват чат интерфейс, който включва списък с чатове, поле за въвеждане на съобщения и секция за показване на съобщения, интерфейс за потребителите, включващ регистрация, вход, управление на приятелства и профил на потребителя и страница за обновяване в реално време, която ще осигурява синхронизация на съобщенията между потребителите в реално време.

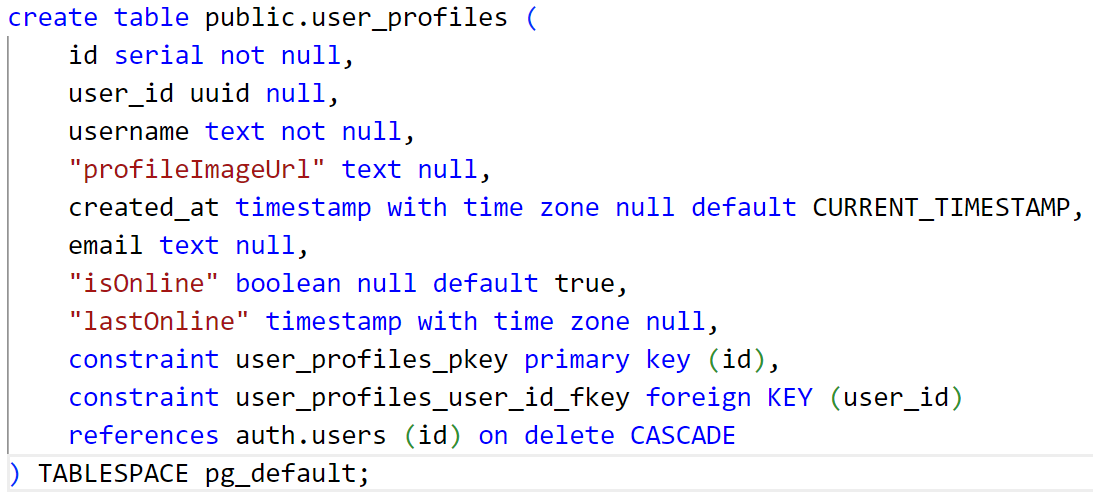
Backend (Сървър) частта ще използва Supabase, който предоставя база данни, автентикация и синхронизация в реално време. Базата данни ще съхранява информация за потребителите, съобщенията, поканите за приятелства и състоянието на всеки чат. Supabase Realtime ще се използва за автоматично обновяване на съобщенията между потребителите, което е основна част за поддържане на активни чатове. Supabase Auth ще се грижи за автентикацията на потребителите чрез имейл и парола, осигурявайки сигурност на платформата.

API Layer ще служи като комуникация между frontend и backend частите на приложението. В този слой ще се изградят основни функции като регистрация и вход на потребителите, изпращане на съобщения и управление на приятелства.

Хостинг ще се осъществява чрез платформа като Vercel, която позволява лесно качване на приложението в интернет и го прави достъпно за потребителите. Платформата е оптимизирана за Next.js, което гарантира бързо и ефективно хостване на проекта.

### Диаграма на базата данни

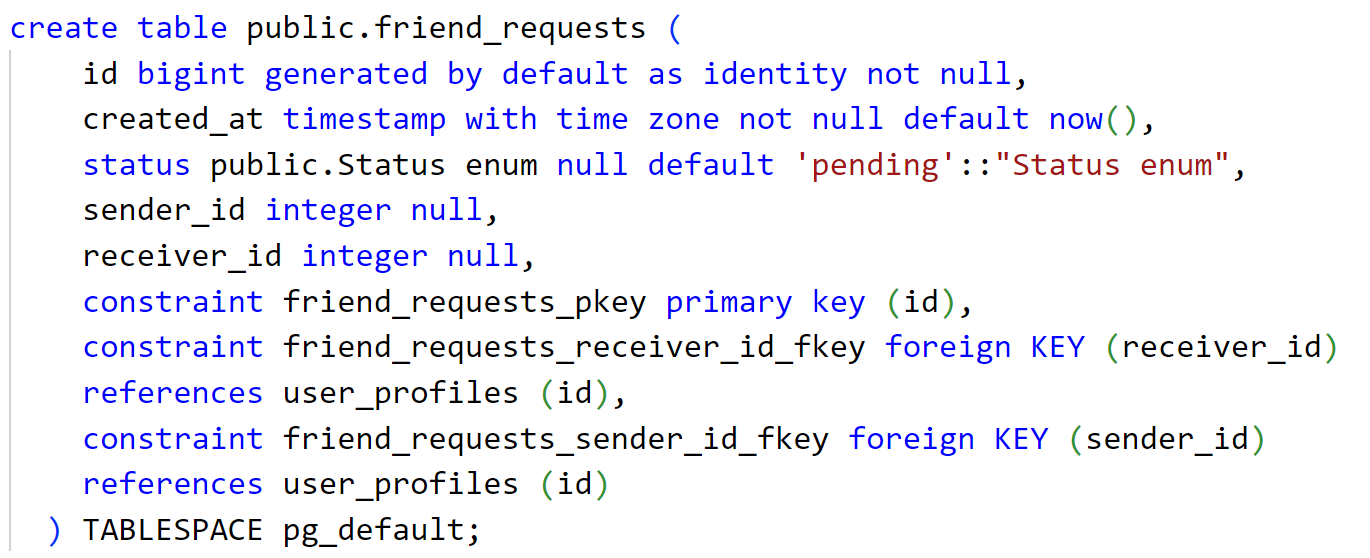
Таблицата user\_profiles съхранява данни за потребителските профили. Полето id е уникален идентификатор за профила и е първичен ключ. User\_id е свързано с уникален потребител от таблицата auth.users. Username е потребителското име и е задължително. profileImageUrl съдържа URL на профилната снимка, което не е задължително, но е пожелателно. Created\_at указва времето на създаване на профила, като по подразбиране е текущото време. Email е имейл адресът на потребителя и също не е задължително поле. Полето isOnline показва дали потребителят е онлайн, като по подразбиране е true. LastOnline съдържа времето на последно онлайн присъствие и не е задължително. Повечето от тези полета като user\_id, profileImageUrl и email може да не са задължителни, но когато се попълват данните в приложението, няма как да останат празни в базата данни.

[](#Ф3121) Таблицата има две ограничения: първичен ключ за id и външен ключ за user\_id, който се свързва с таблицата auth.users и изтрива профила автоматично при изтриване на потребителя.

Фигура ..: Дефиниция на таблица за потребителски профили

Таблицата friend\_requests съхранява заявките за приятелство между потребители. Полето id е уникален идентификатор за всяка заявка и е първичен ключ. created\_at указва времето на създаване на заявката и по подразбиране приема текущото време.

Колоната status представлява състоянието на заявката и използва Status enum, който може да съдържа стойности като `pending`, `accepted` и `declined`. По подразбиране заявката започва със статус `pending`, тъй като очаква действие от получателя – той може да я приеме или откаже.

Полетата sender\_id и receiver\_id указват съответно кой е изпратил и кой е получил заявката. Те са външни ключове, свързани с таблицата user\_profiles, което гарантира, че само съществуващи потребители могат да изпращат и получават заявки. 

Фигура .: Дефиниция на таблица за предлагане на приятелства

Таблицата chat\_room съхранява информация за личните чатове между двама потребители. Полето id е уникален идентификатор за всеки чат и е първичен ключ. created\_at указва времето на създаване на чата и по подразбиране приема текущото време.

Полетата user1 и user2 представляват участниците в чата. Те са външни ключове, свързани с таблицата user\_profiles, което гарантира, че само съществуващи потребители могат да бъдат част от чат.

Тази таблица осигурява структура за съхранение на разговори между двама потребители, като всеки ред представлява един уникален чат между тях.

A screenshot of a computer code

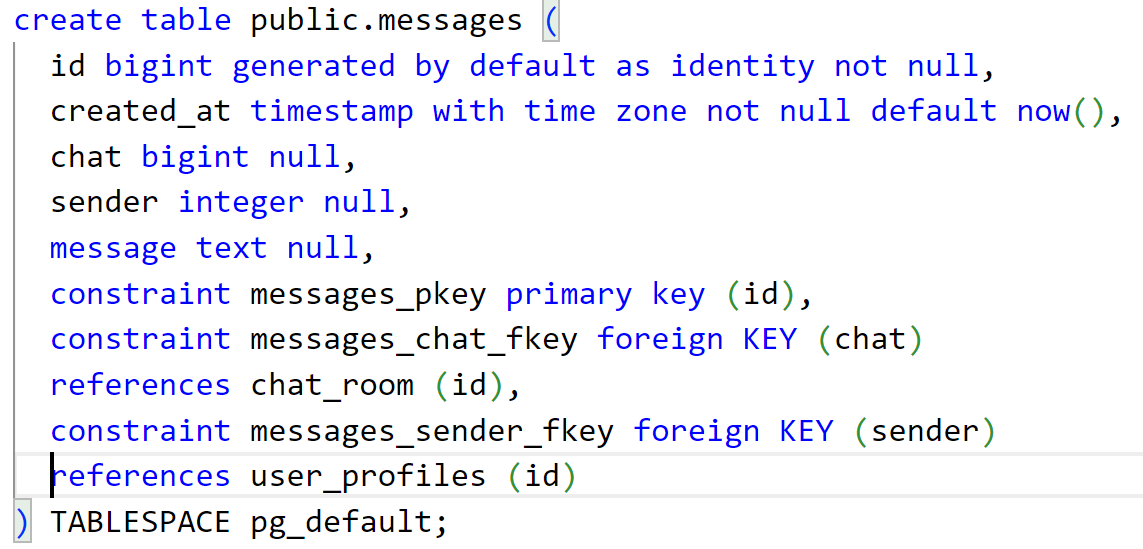
AI-generated content may be incorrect.

Фигура ..: Дефиниция на таблица за чат стаи

Таблицата messages съхранява съобщенията, изпратени в чат стаите. Полето id е уникален идентификатор за всяко съобщение и служи като първичен ключ. created\_at записва времето на изпращане на съобщението и по подразбиране приема текущото време.

chat указва в кой чат е изпратено съобщението и е външен ключ, свързан с chat\_room(id), което гарантира, че всяко съобщение принадлежи на валидна чат стая. sender определя кой потребител е изпратил съобщението и е външен ключ, свързан с user\_profiles(id).

Полето message съдържа съдържанието на самото съобщение, като може да бъде текстово съобщение или друг тип съдържание, в зависимост от разширенията на системата.



Фигура .:Дефиниция на таблица за съобщения

### Диаграма на базата данни

На фигурата може да видим диаграма, тя представлява релационната структура на базата данни за чат система, като визуализира връзките между основните таблици. Тя показва как потребителите комуникират помежду си чрез чат стаи и съобщения, както и как се изпращат покани за приятелство. Всяка таблица има своя роля в цялостното функциониране на системата и е свързана с други таблици чрез външни ключове.

Централна за системата е таблицата user\_profiles, която съдържа информация за всеки потребител, като идентификатор, потребителско име, имейл и статус на активност. Тази таблица е свързана с останалите, тъй като всеки потребител може да изпраща и получава приятелски покани (friend\_requests), да участва в чат стаи (chat\_room) и да изпраща съобщения (messages).

Връзките между потребителите се управляват чрез таблицата friend\_requests, която записва изпратените покани за приятелство и техния статус (изчакваща, приета или отказана). След като двама потребители станат приятели, те могат да започнат чат комуникация чрез таблицата chat\_room, която създава уникална стая за всеки разговор.

Всяка чат стая съдържа множество съобщения, които се записват в таблицата messages. Всяко съобщение има идентификатор, референция към съответната чат стая, изпращач и текстово съдържание. Чрез тази структура се гарантира, че всички съобщения са свързани със съответните разговори и участници.

Диаграмата ясно показва логическите връзки между таблиците и как данните се организират в системата. Чрез използването на външни ключове се гарантира целостта на данните, като например невъзможността да съществува съобщение без валидна чат стая или потребител. Тази организация прави базата данни ефективна и лесна за разширяване с допълнителни функционалности.

A screenshot of a computer

AI-generated content may be incorrect.

Фигура .: Диаграма на базата данни

### Създаване на приложение

A computer screen shot of a computer code

AI-generated content may be incorrect. За да създадем проекта първо отваряме своята среда за разработка. За текущия проект ще използвам Visual Studio code. Когато отворим своята среда, отваряме терминала и пишем командата **„npx create-next-app@latest chat-app”. npx** е инструмент от Node.js, който позволява да инсталираме пакети директно, без да ги изтегляме глобално. **create-next-app@latest** е официална команда, позволяваща автоматичното създаване на проект с всички необходими файлове и настройки, докато **@latest** означава че ще използваме последната и най-нова версия за нашето приложение. **Chat-app** e името на папката, в която се намира всичко. Когато се напише цялата команда ще започнат да се теглят пакетите. След това ще видим допълнителни въпроси.

Фигура ..:Избор при създаване на уеб приложение

За този проект няма да използвам TypeScript, тъй като JavaScript е по-лесен за започване и не изисква допълнителна конфигурация. Това улеснява бързото разработване, без да се налага да дефинираме типове за всяка променлива или функция.

Избрах да използвам ESLint, защото той ще помага за поддържането на чист и подреден код. Този инструмент открива синтактични грешки, несъответствия в стила и други потенциални проблеми, което ще подобри качеството и четимостта на кода по време на разработката.

Tailwind CSS е много важно да го включим. Това е библиотека, която ще помогне за бързото писане и стилизиране на проекта. Tailwind ускорява разработката на UI, като позволява бързо стилизиране с малко код.

Файловете на проекта ще бъдат организирани в папка src/, което е стандартна практика за по-добра структура и разделяне на кода. Това позволява по-лесно управление на компонентите и страниците в приложението.

Избрах да използвам App Router, който е новата система за маршрутизация в Next.js. Вместо класическата pages/ директория, app/ позволява по-гъвкаво управление на сървърни компоненти, подобрено кеширане и по-добра производителност. Така ще си осигурим повече контрол върху структурата и сървърните компоненти на приложението.

Turbopack е другото нещо, което ще вземем под внимание. Не го включвам, защото той все още е в експериментален стадий. Въпреки че обещава значително по-бърза компилация в сравнение с Webpack, засега предпочитам да използвам стабилното решение на Webpack, което е тествано и широко използвано.

Последното нещо е как искаме да импортваме файлове. В Next.js това става по подразбиране с **@/**, това улеснява пътя. Вместо да пишем точки, в зависимост от местоположението на файла само въвеждаме @/ това ни праща в най-горната папка. От там може по лесно да навигираме до файла, който търсим.

С тези настройки проектът е подготвен за работа с Next.js, ESLint и Tailwind CSS, като кодът ще бъде организиран в src/ директория. Използваме App Router за по-ефективна маршрутизация и Webpack като стабилно решение. След приключване на конфигурацията, можем да продължим с разработката на функционалностите на приложението.

### Структура на проекта и основни файлове

В нашия новосъздаден проект ще видим следните файлове:

node\_modules – Съдържа всички инсталирани зависимости (пакети), необходими за проекта.

public – Папка за статични файлове като изображения, шрифтове и икони. Файловете тук се достъпват директно чрез URL.

src/ – Основната директория за кода на приложението.

Вътре в src/app/:

layout.js – Дефинира общото оформление (layout) на страниците. Използва се за споделени елементи като хедър и футър.

page.js – Главната начална страница на приложението.

Конфигурационни и системни файлове

favicon.ico – Иконата на сайта, която се показва в браузъра.

globals.css – Глобален CSS файл, където се задават основните стилове за целия проект.

.gitignore – Определя кои файлове и папки да бъдат игнорирани от Git (например node\_modules).

eslint.config.mjs – Конфигурация за ESLint, който анализира кода и помага да се поддържат добри практики.

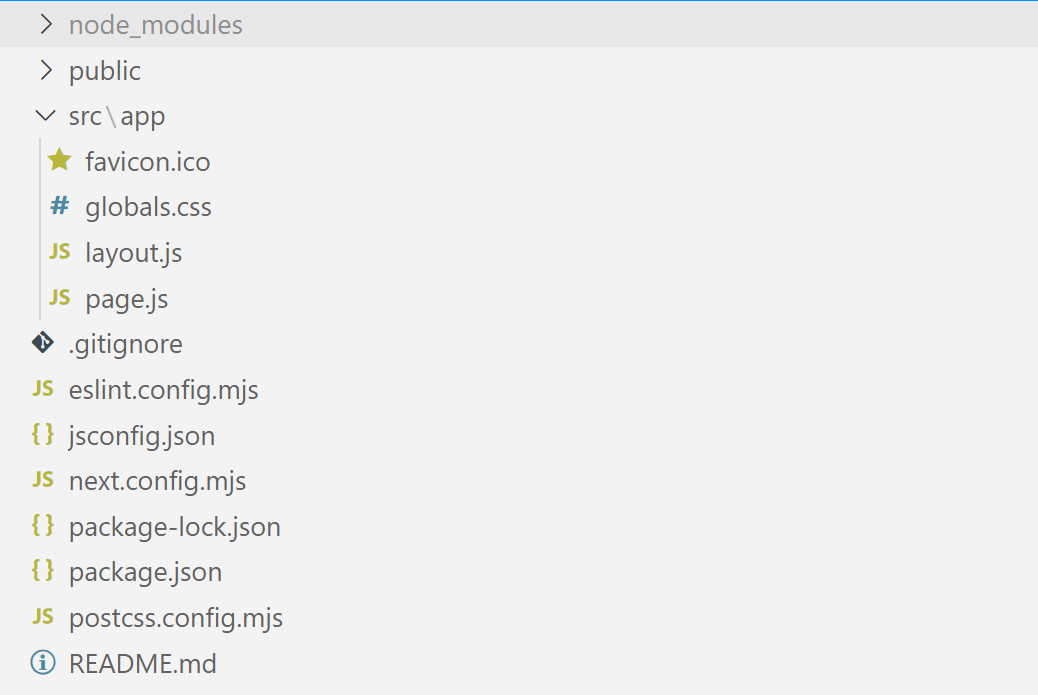
jsconfig.json – Конфигурация за JavaScript, която улеснява импортите и работата с пътищата в проекта.

next.config.mjs – Основният конфигурационен файл за Next.js, където могат да се зададат настройки като оптимизация на изображения и API маршрути.

package.json – Описва проекта, съдържа информация за зависимостите и командите за изпълнение (скриптове).

package-lock.json – Заключва версиите на зависимостите, за да се гарантира, че всички в екипа използват една и съща версия на пакетите.

postcss.config.mjs – Конфигурация за PostCSS, който се използва за обработка на CSS (необходим за Tailwind CSS).

README.md – Документ с инструкции и описание на проекта.

Фигура .: Папки и файлове в новосъздаден проект

### Изтегляне на пакети

A black and blue text

AI-generated content may be incorrect. За текущия проект ни трябва само един допълнителен пакет и той е библиотеката на Supabase. **@supabase/supabase-js** е официалната JavaScript библиотека (SDK) с която ще работим. С нея можем да се свързваме с базата данни, да пращаме заявки за четене и писане, да работим в реално време и много други.

Фигура .:Команда за изтегляне на пакет

## Създаване и разпределяне на папки

Има няколко важни папки, които да си създадем преди да започнем самото писане на код за проекта.

### Файл **.env**

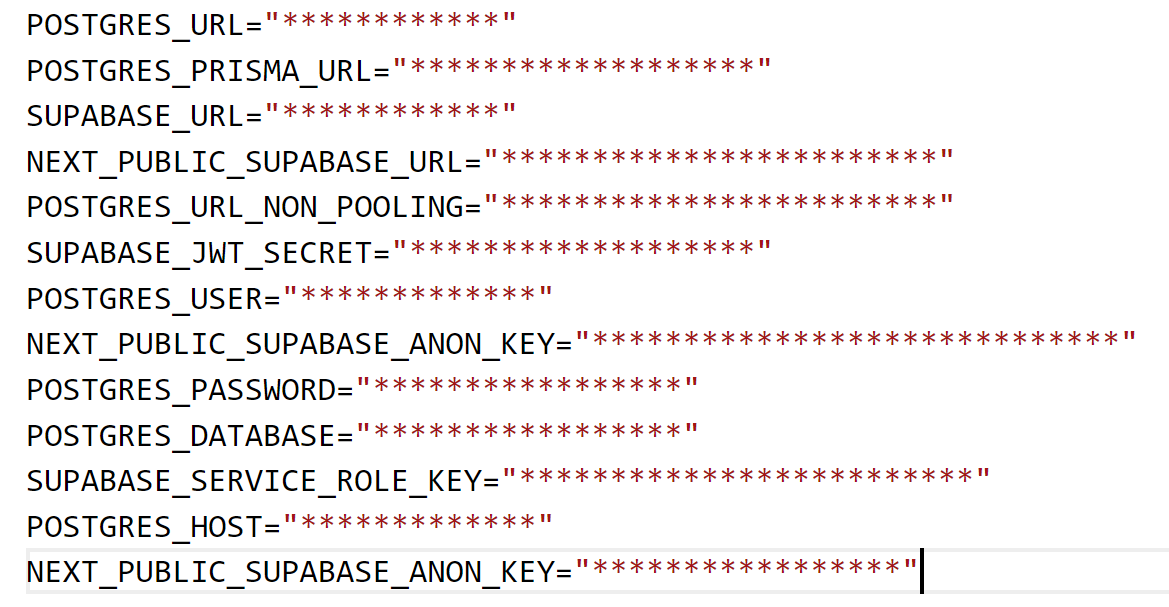
Файлът .env (от англ. "environment") е специален текстов файл, който се използва за съхраняване на околните променливи в проектите. Околните променливи са стойности, които дефинират конфигурацията на приложението и могат да включват чувствителни данни, като API ключове, потребителски имена, пароли, токени за автентикация и други важни данни, които не трябва да бъдат споделяни публично.

Значение на .env файла:

Сигурност и конфиденциалност на данните: Основната причина за използването на .env файл е, че той предоставя начин за безопасно съхранение на чувствителни данни. Данни като API ключове и пароли не трябва да бъдат включвани директно в изходния код на проекта, особено когато проектът е публикуван на публични репозитории, например в GitHub. С .env файла тези данни се държат локално на машината или сървъра и не се излагат на външни лица.

Гъвкавост и конфигурируемост: Чрез използването на .env файл, разработчиците могат да дефинират различни стойности за различни среди. Например, едни стойности могат да бъдат използвани в локалната среда на разработчика, други в тестовата среда, а трети в продукционната среда. Това позволява на проектите да бъдат по-гъвкави и лесно конфигурируеми в зависимост от нуждите на съответната среда, без да е необходимо да се променя кодът.

Лесно управление на конфигурациите: Използването на .env файл улеснява конфигурирането на проектите, като същевременно намалява риска от грешки. Вместо да въвеждат ръчно чувствителни данни на различни места в кода, разработчиците могат просто да добавят стойностите в .env файла и да използват съответните променливи в приложението. Това прави кода по-четим и лесен за поддръжка.

За нашият пример ни трябват най-важната информация, която се отнася до базата ни данни. Нея може да я намерим в Vercel, мястото което ще хоства нашия проект. 

Фигура ..: .env файла

Така трябва да изглежда нашият файл, разбира се без звездичките, но няма да разкривам ценна информация в проекта.

### Папката lib

Папката lib е другата много важна. В нея ще държим всички файлове, свързани с базата данни и работата с профили. Един от важните файлове в нея е **setUp.js.** Това е файлът, в който инициализираме базата.

import { createClient } from '@supabase/supabase-js'

A close up of a computer code

AI-generated content may be incorrect.Тази линия импортира функцията createClient от библиотеката @supabase/supabase-js, която е JavaScript SDK на Supabase. Тази функция се използва за създаване на клиент, който ще се свърже с Supabase базата данни.

Фигура ..: Инициализиране на базата данни

export const supabase = createClient(process.env.NEXT\_PUBLIC\_SUPABASE\_URL, process.env.NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY);

Тази линия създава клиентски обект supabase, който се свързва със Supabase проекта, използвайки стойности за URL и анонимния ключ на проекта.

process.env.NEXT\_PUBLIC\_SUPABASE\_URL:

Това е URL адресът на нашия Supabase проект. Той се съхранява в променлива на средата и може да бъде достъпен чрез process.env.

process.env.NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY

Това е анонимният ключ за достъп до проекта, който позволява на клиентите да изпълняват основни операции без нужда от автентикация. Този ключ също е съхранен в променлива на средата.

### Папката UI

Папката UI (потребителски интерфейс) съдържа всички компоненти и ресурси, които са отговорни за визуализацията и взаимодействието с потребителя в проекта. Тя обикновено включва файлове като:

React компоненти: Това са отделни части от интерфейса, като бутони, форми, навигационни панели и други елементи, които се използват многократно в приложението.

CSS/Tailwind CSS класове: Стилове, които дефинират визуалната част на компонентите, включително оформление, цветове, шрифтове и анимации.

Изображения и икони: Ресурси, които са необходими за графичната част на интерфейса, като икони, лога и изображения, използвани в компоненти.

Целта на папката UI е да предостави лесен начин за управление на всички визуални компоненти, като по този начин се улеснява поддръжката и разширяването на интерфейса на проекта. Това позволява централизирано управление на визуалните елементи и улеснява тяхното повторно използване в различни части на приложението.

### Папката Hooks

Папката Hooks съдържа персонализирани React hook-ове, които капсулират логика за многократна употреба. В настоящия проект в тази папка се намира hook-ът useAuth, който отговаря за управлението на автентикацията на потребителите. Той обединява логиката за вход, изход и следене на текущата сесия на потребителя чрез Supabase. С използването на useAuth се улеснява достъпът до информация за потребителя във всички компоненти на приложението и се осигурява по-чист и структуриран код.

### Папката Utils

Папката Utils съдържа помощни функции и скриптове, които подпомагат основната логика на приложението. Те не са свързани директно с визуалната част или с конкретни компоненти, а изпълняват отделни логически задачи, които се използват на различни места в проекта. Извеждането на такава логика в отделни файлове улеснява повторната ѝ употреба, прави кода по-организиран и по-лесен за поддръжка.

В тази папка се намират функции, свързани с обработка на пътища, локално съхранение на данни и форматиране на време. Благодарение на папката utils, проектът е по-модулен, а логиката е отделена от интерфейсната част, което улеснява и бъдещи разширения и промени.

### Папката Context

Папката Context съдържа глобални състояния и логика за споделяне на данни между различни компоненти в приложението чрез React Context API. Основната ѝ цел е да улесни достъпа до определени данни – като информация за текущия потребител – без да е необходимо те да се предават ръчно през всяко ниво на компоненти (props). Така се осигурява по-ефективно управление на състоянието и по-добра структура на проекта, особено при по-големи приложения с много взаимосвързани елементи.