**A logo with black text

Description automatically generated Технически университет - София**

Дисциплина: „Софтуерни системи за управление на бизнеса“

**Техническа спецификация**

**Тема: „Web базирано приложение за осиновяване на домашни любимци Adoptly"**

София

2025г.

Съдържание

[**1.** **Въведение** 3](#_Toc198652584)

[**2.** **Технологичен стек** 4](#_Toc198652585)

[**2.1.** **Преглед на използваните технологии** 4](#_Toc198652586)

[**3.** **Системна архитектура** 9](#_Toc198652587)

[**3.1.** **Основни аспекти на системната архитектура** 10](#_Toc198652588)

[**3.2.** **Системна архитектура на проекта Adoptly** 11](#_Toc198652589)

[**4.** **База данни и моделиране** 13](#_Toc198652590)

[**4.1.** **Модел на базата данни** 13](#_Toc198652591)

[**5.** **Моделиране на системата** 15](#_Toc198652592)

# **Въведение**

Проблемът с бездомните животни в България е комплексен и за него все още няма ясно решение. Всеки ден се сблъскваме с изоставени животни на улицата, които са оставени сами да се борят за своето оцеляване. Тази ситуация не само причинява огромни страдания на животните, но и представлява сериозна заплаха за общественото здраве и безопасност. Липсата на адекватни мерки и ресурси за защита на техния живот и здраве само влошава ситуацията, създавайки опасност както за животните, така и за хората около тях. Разбира се, съществуват и много отговорни граждани, които са готови да предоставят дом на животно в нужда, временно или за постоянно, като им осигурят грижа и подслон. За съжаление, липсата на ефективна платформа или система, която да осъществи връзка между тези, които търсят нов любимец, и онези, които дават животни за осиновяване, води до сериозни пропуски в процеса. Тази липса на координация не само затруднява намирането на подходящи осиновители, но и допринася за увеличаване на броя на бездомните животни. В резултат на това, дори и с наличието на хора, готови да помогнат, много животни остават на улицата, където продължават да страдат и да създават допълнителни проблеми за обществото.

Настоящият проект предлага разрешение чрез разработването на уеб приложение на име Adoptly, което ще служи като платформа за свързване на хора, които искат да дадат животни за осиновяване, с тези, които търсят да предоставят нов дом, изпълнен с любов и грижа за тях. Приложението ще създаде лесен и достъпен начин за комуникация между двете страни, улеснявайки процеса на осиновяване и гарантирайки, че животните ще намерят подходящи семейства, които да се грижат за тях.

В заключение, разработването на това приложение има потенциала да окаже положителен ефект както за бездомните животни, така и за обществото като цяло. Чрез улесняването на процеса на осиновяване, то ще насърчи по-голяма ангажираност и отговорност сред гражданите и ще допринесе за намаляване на броя на бездомните животни.

# **Технологичен стек**

## **Преглед на използваните технологии**

* **Node.js:**

Безплатна крос-платформена среда с отворен код, на която може да се изпълнява JavaScript код (т.нар. runtime environment)  директно на сървъри. Изградена е върху V8  Javascript Google engine и се изпълнява извън браузъра, което прави продуктивността й голяма и я превръща в популярен избор за разработка на сървърни приложения, уеб услуги и инструменти за команден ред.

* **JavaScript:**

JavaScript е универсален, интерпретируем програмен език, широко използван за създаване на интерактивни уеб страници и сложни мобилни, уеб и сървърни приложения. Появата на Node.js го превърна в универсален език за разработка, използван както за фронтенд, така и за бекенд. Вграден във всички съвременни уеб браузъри, JavaScript е един от най-популярните езици за уеб програмиране, благодарение на своята гъвкавост, богата екосистема от библиотеки и фреймуърк като React, Angular и Vue.js, както и активната общност от разработчици.

* **TypeScript:**

TypeScript е език за програмиране, който разширява синтаксиса на JavaScript, добавяйки статична типизация, което улеснява изграждането на сложни и мащабируеми уеб приложения. Основната му характеристика е статичната типизация, която позволява на разработчиците да дефинират типове за променливи, функции, обекти и класове, като по този начин се избягват много от грешките, възникващи при изпълнението на код без типове, както в JavaScript. Това води до по-стабилен код и намалява времето за откриване и коригиране на грешки. Езикът е съвместим с всички функционалности на JavaScript, което позволява лесна миграция на съществуващи проекти към TypeScript, без да се налага радикална промяна в кода. Освен това, той се компилира в стандартен JavaScript, което означава, че може да бъде изпълняван в браузъри, сървъри и всякакви платформи, поддържащи JavaScript.

* **React:**

Популярна библиотека с отворен код за изграждане на потребителски интерфейси, разработена от Facebook. Основната й цел е да улесни разработката на динамични и интерактивни уеб приложения, като позволява на разработчиците да създават компоненти – малки, независими парчета код на JavaScript, които могат да се преизползват. React използва т.нар. „виртуален DOM“ (Document Object Model) [5], който значително подобрява производителността намалявайки броя на промените в реалния DOM. Това прави React особено подходящ за изграждане на мащабируеми и високоефективни уеб приложения. Тази библиотеката се интегрира лесно с други технологии и се използва широко в комбинация с Node.js за създаване на интерактивни уеб приложения.

* **NextJS:**

Next.js е фреймуърк (на англ. framework) с отворен код, базиран на React, който улеснява разработката на уеб приложения с функционалности като сървърно рендериране и статично генериране на страници за по-висока производителност. Разработен от Vercel, той автоматично управлява маршрути и предлага вградена поддръжка за API маршрути, което позволява създаване на бекенд функционалности директно в приложението.

* **HTML/CSS:**

HTML (HyperText Markup Language) и CSS (Cascading Style Sheets) вървят ръка за ръка и са основните градивни елементи на уеб разработката. HTML служи за създаване на структурата и организацията на съдържанието в интернет, докато CSS е стиловият език, който определя външния вид и оформление на това съдържание. Заедно, HTML и CSS създават основата на всяка уеб страница, осигурявайки както функционалност, така и визия.

* **Tailwind:**

Tailwind CSS е CSS фреймуърк, който позволява  бързо и ефективно създаване на персонализиран дизайн чрез използване на готови класове директно в HTML. Той предоставя висока гъвкавост и контрол върху стилизирането, без да се налага писането на традиционен CSS код. Tailwind е особено популярен заради своята ефективност и възможността за лесно преизползване на стилове.

* **Shadcn:**

Колекция от готови за използване компоненти, изградени върху Radix UI [10] и Tailwind CSS, която улеснява създаването на потребителски интерфейс за уеб приложения. Предоставя предварително стилизирани и функционални компоненти, които са лесни за интеграция и персонализация. Shadcn комбинира мощността на Tailwind CSS за стилове с функционалността на Radix UI, осигурявайки гъвкавост и контрол върху дизайна. Предназначена е да бъде лесна за надграждане, което позволява на разработчиците да адаптират компонентите към специфичните нужди на проекта. Тази колекция от компоненти значително ускорява процеса на разработка, като също така гарантира, че крайният резултат е визуално привлекателен и лесен за поддръжка.

* **MongoDB:**

Нерелационна база данни (или NoSQL база данни) с отворен код, която е алтернатива на традиционните релационни бази данни. Вместо да използва таблици и редове, MongoDB съхранява данни в документи, които са структурирани в JSON-подобен формат, наречен BSON (Binary JSON). Всеки документ представлява отделен запис и съдържа ключ-стойност двойки, което позволява съхранението на различни типове данни в един и същ документ. Лесна е за интеграция и предлага функции като автоматично разпределение на данните и хоризонтално мащабиране. Предлага също и възможност за ефективно управление на динамични схеми, което позволява разработчиците бързо да адаптират структурата на данните спрямо променящите се нуждите на проекта.

* **Mongoose:**

Библиотека за Node.js, която служи като обвивка (ORM - Object-Relational Mapping) за работа с MongoDB бази данни. Основното й предимство е, че позволява създаването на модели (схеми), които дефинират как са представени данните в колекциите. Моделите също така включват валидации, дефинирани типове данни и виртуални полета, които улесняват работата с данни.

* **NextAuth.js:**

Библиотека за удостоверяване (authentication) с отворен код, предназначена за приложения,  базирани на Next.js. Предлага лесен и гъвкав начин за интегриране на различни методи за автентикация, като социално влизане чрез Google, Facebook, GitHub, както и стандартния метод с имейл и парола. Едно от основните й предимства на е лесната интеграция с Next.js, което позволява на разработчиците да добавят автентикация с минимални усилия. Библиотеката поддържа както сървърно рендиране, така и API маршрути, което осигурява допълнителна сигурност и спазване на добрите практики за уеб разработка.Освен това NextAuth.js предлага функции като управление на сесии, защита на страници и лесна персонализация на логиката за влизане и регистрация. Тези характеристики я популярна сред Next.js разработчиците заради своята гъвкавост и възможност за бързо внедряване на сигурно удостоверяване в уеб проекти.

* **Zod:**

Лека библиотека за валидиране на данни и дефиниране на схеми в TypeScript и JavaScript. Тя позволява лесно създаване на схеми за данни, като осигурява автоматична интеграция с TypeScript типизацията. Тя предоставя вградени валидатори за основни типове данни и позволява комбиниране на различни схеми за по-сложни структури. Освен това, библиотеката е известна с опростения си синтаксис и лесната обработка на грешки.

* **Bcrypt.js**:

Представлява JavaScript библиотека, която реализира алгоритъма Bcrypt за хеширане на пароли. Тя осигурява сигурен начин за защита на пароли чрез добавяне на "salt" и настройване на сложността на хеширане (cost factor), правейки паролите по-трудни за разбиване. Bcrypt.js е лесна за използване и широко приложима в Node.js приложения за осигуряване на надеждна защита на потребителските данни.

Всички тези технологии работят заедно, за да създадат стабилна, гъвкава и мащабируема платформа, която улеснява процесите по осиновяване на животни. Те предоставят надеждно и сигурно решение, което позволява на разработчиците да персонализират и адаптират приложението според нуждите на проекта.

# **Системна архитектура**

Създаването на ефективно софтуерно решение за справяне с проблема с бездомните животни изисква креативност и прецизно планиране. В тази глава ще се разгледа  процеса на проектиране на интуитивен потребителски интерфейс и надеждна системна архитектура. Целта ни е да разработим приложение, което не само ще улесни хората, които искат да осиновят или помогнат на животните в нужда, но и ще насърчи повече хора да се ангажират активно с решаването на този важен социален въпрос.

* **Потребителите :**

Фокусирането върху лесния и удобен достъп до информацията за животните е ключово за разработването на приложението Adoptly. Това е постигнато чрез предоставяне на възможност за директно преглеждане на обяви през потребителския интерфейс, без нужда от сложни стъпки.

Интерфейсът е проектиран така, че функциите за преглед и публикуване на обяви са достъпни с няколко клика, което позволява бързо и ефективно намиране на животни за осиновяване.

Важно предимство на приложението е, че не изисква сложни регистрации или специфични акаунти, което разширява обхвата на потребителите и улеснява достъпа до информацията. С интуитивния си дизайн, Adoptly прави процесите на осиновяване и публикуване на обяви лесни и достъпни, като подобрява потребителското изживяване и ангажираността в каузата за намаляване на бездомните животни.

## **Основни аспекти на системната архитектура**

За създаването на ефективно приложение за осиновяване на животни, базирано на Next.js, е необходимо задълбочено разбиране на архитектурните принципи на тази платформа и свързаните технологии. Този фреймуърк осигурява основната структура за разработка на съвременни уеб приложения, предлагайки функционалности като сървърно рендериране и статично генериране на страници, които са от съществено значение за гарантиране на висока производителност и SEO оптимизация на крайния продукт. Това прави Next.js идеален избор за изграждане на интегрирана система, която да съчетава както фронтенд, така и бекенд функционалности в едно цялостно решение.

Архитектурата на приложението Adoptly включва няколко ключови компонента, които работят в синхрон за осигуряване на оптимално потребителско изживяване и сигурност на данните. Основните технологии, използвани в проекта, са:

* **Next.js за фронтенд и бекенд:**

Next.js управлява потребителския интерфейс и сървърните заявки, осигурявайки висока производителност и SEO оптимизация чрез сървърно рендериране и статично генериране на страници.

* **MongoDB за съхранение на данни:**

MongoDB съхранява потребителска информация и данни за животни, осигурявайки гъвкавост и ефективност при работа с документо-ориентирани структури.

* **NextAuth.js за автентикация и управление на сесии:**

NextAuth.js осигурява сигурна автентикация и управление на потребителските сесии с поддръжка на различни методи за вход.

## **Системна архитектура на проекта Adoptly**

Разработването на приложението Adoptly изисква интеграция на няколко ключови технологии, които да осигурят сигурност, ефективност и лесна употреба. Системната архитектура на Adoptly е създадена с цел да улесни процеса на осиновяване на животни, като използва Next.js за фронтенд и бекенд, MongoDB за съхранение на данни и NextAuth.js за автентикация на потребителите.

* **Настройки на потребителски профили и автентикация:**

В Adoptly автентикацията е централизирана чрез NextAuth.js, като позволява влизане с имейл, парола или чрез OAuth доставчици като Google. Профилите и сесиите се управляват с JWT токени и се съхраняват в MongoDB.

* **Създаване на обяви за осиновяване:**

Потребителите създават обяви с информация за животните, които се съхраняват в MongoDB. Обявите могат да се преглеждат и управляват през потребителските профили, а други потребители могат да се свързват с лицата, които предлагат животните за осиновяване *фиг. 1.*

* **Достъп до информация и актуализации:**

Информацията за животните и обявите се съхранява в MongoDB, като потребителите могат да я достъпят през Adoptly. След влизане, те могат да разглеждат свои и чужди обяви, като всички заявки към MongoDB се управляват чрез API маршрути на Next.js, осигурявайки лесен достъп и управление на данните.

*Картина, която съдържа текст, диаграма, скица, Техническо чертане

Генерираното от ИИ съдържание може да е неправилно.*

*Фигура 1 :  Блок схема на процеса по създаването на обява*

# **База данни и моделиране**

# **Модел на базата данни**

На *фиг.3* се илюстрира моделът на  базата данни на Adoptly. Тази диаграма показва начинът, по който се запазва информацията в MongoDB- колекциите с данни и връзките и взаимодействията между тях.

На приложената схема са изобразени връзките между следните таблици:

* **User**: регистриран потребител в приложението
  + id- първичен ключ в GUID формат
  + username- потребителското име на потребителя.
  + email- електронна поща на потребителя, с която се е регистрирал.
  + password- паролата за регистрацията. Всяка една парола се хешира и резултатът се запазва в базата данни.
  + img- профилна снимка на потребителя. Полето може да приема NULL стойности за потребителите, които не биха искали профилът им да има изображение.
  + isAdmin- флаг който показва дали даден потребител е администратор.
* **Animal**: регистрирано животно в приложението.
  + id- първичен ключ в GUID формат.
  + description- информация в свободен текст за самото животно.
  + type- видът на животното.
  + age- възрастта на животното.
  + gender- полът на животното.
  + userId- външен ключ към записи в таблицата User. Връзката е 1:много, тъй като един потребител може да има много животни а едно животно може да принадлежи само на един потребител.
* **Photo:** снимките в приложението.
  + id- първичен ключ в GUID формат.
  + title- заглавието на снимката.
  + src- Не се запазват самите снимки в базата данни а пътя до тях. Връзката е 1:много, тъй като едно животно може да има много снимки а една снимка може да принадлежи само на едно животно.
  + animalId- външен  ключ към записи в таблицата Animal.

Картина, която съдържа текст, екранна снимка, номер

Генерираното от ИИ съдържание може да е неправилно.

*Фигура 3 : Модел на базата данни на приложението.*

# **Моделиране на системата**

В този раздел са представени основните UML диаграми, които описват архитектурата, функционалността и взаимодействията в системата.

Картина, която съдържа текст, екранна снимка, диаграма, Лепяща бележка

Генерираното от ИИ съдържание може да е неправилно.

*Фигура 4: Инсталационна диаграма*

*Картина, която съдържа текст, диаграма, екранна снимка, Правоъгълник

Генерираното от ИИ съдържание може да е неправилно.*

*Фигура 5: Компонентна диаграма*

*Картина, която съдържа текст, екранна снимка, диаграма, Правоъгълник

Генерираното от ИИ съдържание може да е неправилно.*

*Фигура 6: Timing диаграма за вход в системата*

*Картина, която съдържа диаграма, текст, екранна снимка, Правоъгълник

Генерираното от ИИ съдържание може да е неправилно.*

*Фигура 7: Timing диаграма за регистрация в системата*

*Картина, която съдържа текст, диаграма, екранна снимка, Паралелен

Генерираното от ИИ съдържание може да е неправилно.*

*Фигура 8: Sequence диаграма за вход в системата*

*Картина, която съдържа текст, Паралелен, екранна снимка, диаграма

Генерираното от ИИ съдържание може да е неправилно.*

*Фигура 9: Sequence диаграма за регистрация в системата*

*Картина, която съдържа текст, диаграма, линия, бял

Генерираното от ИИ съдържание може да е неправилно.*

*Фигура 10: Use-case диаграма*