**ДИПЛОМЕН ПРОЕКТ**

**ЗА ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ**

по професия код 481030 „Приложен програмист“

специалност код 4810301 Приложно програмиране“

ТЕМА: „ПРИЛОЖЕНИЕ ЗА УПРАВЛЕНИЕ НА ЗАДАЧИ И РАБОТА“

Автор: Йоанна Владимирова Симеонова, XII „В“

Ръководител: Виктор Стоев

БургасСЪДЪРЖАНИЕ

[1 Увод 3](#_Toc155035766)

[2 Цели и обхват на софтуерното приложение 3](#_Toc155035767)

[3 Анализ на решението 4](#_Toc155035768)

[3.1 Потребителски изисквания и работен процес 4](#_Toc155035769)

[3.2 Примерен потребителски интерфейс 5](#_Toc155035770)

[3.3 Диаграми на анализа 5](#_Toc155035771)

[3.4 Модел на съдържанието / данните 5](#_Toc155035772)

[4 Дизайн 6](#_Toc155035773)

[4.1 Реализация на архитектурата на приложението 6](#_Toc155035774)

[4.2 Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой. 6](#_Toc155035775)

[4.3 Организация и код на заявките към база от данни 6](#_Toc155035776)

[4.4 Наличие на потребителски интерфейс (конзолен, графичен, уеб) 6](#_Toc155035777)

[5 Ефективност и бързодействие на решението 7](#_Toc155035778)

[6 Тестване 7](#_Toc155035779)

[7 Заключение и възможно бъдещо развитие 7](#_Toc155035780)

[8 Използвани литературни източници и Уеб сайтове 7](#_Toc155035781)

[9 Приложения 8](#_Toc155035782)

[10 Критерии и показатели за оценяване 9](#_Toc155035783)

# Увод

В съвременната динамична работна среда управлението на задачи и проекти е от съществено значение за ефективността и успеха на организациите. С нарастването на дистанционната работа и необходимостта от по-добра координация между екипите, използването на софтуерни решения за управление на задачи става все по-актуално. Това приложение за управление на задачи и работа е разработено с цел да предостави интуитивен и ефективен инструмент за проследяване на задачите, подобряване на организацията и повишаване на продуктивността.

Проектът решава проблема с липсата на централизирана система за управление на задачи, което често води до загуба на време, неефективна комуникация и пропуски в изпълнението на важни задачи. В контекста на съвременните работни процеси, особено в екипите, които работят дистанционно или в хибридна среда, необходимостта от ефективен инструмент за управление на задачи е критична.

За моделиране на архитектурата на приложението е използван езикът UML (Unified Modeling Language) в комбинация със средата за моделиране Lucidchart. Чрез използването на UML диаграми, като диаграми на случаи на употреба, диаграми на класове и диаграми на последователност, е структурирана ясна представа за взаимодействието между различните компоненти на системата. Този подход осигурява лесна разбираемост и поддръжка на проекта.

Останалата част от документацията е структурирана, както следва:

1. Цели и обхват на софтуерното приложение – определяне на обхвата и основните цели на системата.
2. Анализ на решението:
   1. Потребителски изисквания и работен процес.
   2. Примерен потребителски интерфейс.
   3. Диаграми на анализа.
   4. Модел на съдържанието / данните.
3. Дизайн:
   1. Реализация на архитектурата на приложението.
   2. Описание на слоевете, предназначението им, библиотеки и методи, включени в съответния слой.
   3. Организация и код на заявките към база от данни.
   4. Наличие на потребителски интерфейс.
4. Ефективност и бързодействие на решението – анализ на производителността и оптимизациите.
5. Тестване – описание на използваните тестови стратегии и резултати.
6. Заключение и възможно бъдещо развитие – обобщение на постиженията и идеи за усъвършенстване.
7. Използвани литературни източници и уеб сайтове – списък на референциите.
8. Приложения – допълнителни материали, свързани с проекта.

Този документ предоставя цялостен поглед върху процеса на разработка на приложението и неговата роля в подобряването на управлението на задачи и работа.

# Цели и обхват на софтуерното приложение

Идеята за създаване на софтуерното приложение произтича от необходимостта от централизирана платформа за управление на задачи и работа в различни екипни и бизнес среди. Приложението има за цел да подпомогне потребителите в организирането, проследяването и изпълнението на задачи по ефективен и структуриран начин

При разработката на приложението ще бъдат използвани методологиите Agile и Scrum. Agile ще осигури гъвкавост в процеса на разработка, като се фокусира върху адаптивността и непрекъснатото подобряване на софтуера въз основа на обратната връзка от потребителите. Scrum, като конкретна методология в рамките на Agile, ще бъде използвана за управление на екипната работа чрез разделяне на проекта на кратки, управляеми спринтове с ясно дефинирани задачи и цели. Това ще позволи бързо идентифициране и решаване на проблеми, както и ефективно разпределение на ресурсите в екипа.

Обхватът на потребителите включва индивидуални потребители, малки и средни екипи, както и големи организации, които се нуждаят от систематизирано управление на работния процес. Приложението ще поддържа следните основни дейности:

* Създаване и управление на задачи.
* Проследяване на напредъка по задачи и проекти.
* Разпределение на задачите между членовете на екипа.
* Настройка на крайни срокове.
* Добавяне на етикети към задачи.
* Канбан дъска за всеки проект.

На база на обхвата и функционалността, основните цели на приложението са:

* Осигуряване на интуитивен и удобен за ползване интерфейс.
* Подобряване на комуникацията и координацията в екипите.
* Осигуряване на надеждно съхранение и управление на данните.

# Анализ на решението

## Потребителски изисквания и работен процес

Системата може да се използва от различни предприятия или индивидуални лица и екипи. Те използват системата, за да организират работния си процес ефективно и да следят какво са свършипли.

Първо те влизат в системата със своите имейл и парола. След това могат да виждат и да създават проекти, да проверяват крайните срокове на своите задачи в календара, да създават и назначават нови задачи. Също така могат да създават отчети и да редактират личните си профили.

## Примерен потребителски интерфейс

Допълнете резултатите от статистически анализа на проблема, описани в секция 3.1, с фигури на примерен графичен интерфейс /създадени или в самата среда заедно с потоците от събития, или извън нея/.

## Диаграми на анализа

Тук опишете резултата от анализа на проблема с UML диаграми

* ER диаграма на базата данни
* използване на клас диаграми на анализа /с класове със стереотипи/ За по-сложните контролни класове представете диаграми на състоянието /евентуално йерархични/
* диаграми на последователността и на комуникацията

Забележки:

1. В зависимост от спецификата на проекта трябва да бъдат разработени съответно необходимите диаграми.

## Модел на съдържанието / данните

Тук опишете модела на данните/съдържанието - текстово, графично и евентуално аудио/видео съдържание), което ще представите в проекта си. Опишете размера и типа на данните/файловете и начина на кодиране за всеки от ресурсите.

Ако ползвате в проекта си текстово и/или мултимедийно съдържание от различни типове, представете неговата структура, напр. посредством таксономия, типология, онтология или други схеми за представяне на структурата от категории, под-категории, типове и т.н., както и техните взаимовръзки с други категории или типове, напр. географски региони и дялове, области/сегменти от промишлеността, и др.

След структурата опишете и възможните характеристики, атрибути и честота на срещане на всеки един ресурс в съдържанието (категория, тип, екземпляр, връзка/релация и т.н.).

# Дизайн

Проектът е написан на C#, което означава, че е основан върху .NET Core платформата. Използва се версия 8 както за основната платформа, така и за другите библиотеки, произведени от Microsoft и други трети страни. Разделен е на 3 слоя – слой на бизнес логиката, слой за достъп до базата данни и презентационен слой, и 6 проекта:

* Презентационен слой - това е слоят, който комуникира с потребителите и представя данните/съдържанието в удобен и интерактивен формат.
  + AlloK8.PL – MVC проект, в който се задават както пътищата, така и изгледите на приложението.
* Слой за бизнес логиката - това е слоят, който съдържа бизнес логиката на приложението. Той е независим от другите слоеве и определя правилата и ограниченията за работа с данните/съдържанието.
  + AlloK8.BLL – Библиотека от класове, която използва интерфейси и абстракции за взаимодействие с другите слоеве. В папката Identity се съдържат сървисите за потребителите, а в Common за останалите функционалности като задачи, проекти, календар и отчети.
* Слой за данните - представен от следния проект:
  + AlloK8.DAL – Това е слоят за данни, където се намират моделите и контекстът за базата данни. Той използва Entity Framework Core за ORM и SQL Server за релационна база данни.
* В приложението има още два проекта:
  + AlloK8.Common – Това са модели за вход (input models) и изход (output models), които се споделят между контролера и бизнес логиката. Те представят данните/съдържанието в сериализиран формат (JSON).
  + AlloK8.BLL.Tests – Това е проектът, отговарящ за тестването на слоя за бизнес логиката на приложението.

Приложението използва набор от зависимости, за да постигне своите цели, а това са главно:

* Entity Framework Core 7 (EF Core 7) – Рамка, която улеснява достъпа до базата данни. Проекта е направен, чрез така наречения начин на първоначалната кодова структура (Code First Approach). Това означава, че първо са писани моделите, с които работи базата данни, а после са преобразувани в таблици със съответните им връзки, отношения и типове на колони.
* SendGrid – Библиотека, която позволява изпращането на имейли от приложението към потребителите.
* xUnit – Рамката, с която е реализирано единичното тестване на слоевете за бизнес логика и достъп до базата данни.
* Stylecop Analyzers – Рамка, която анализира C# кода, за да наложи набор от правила за стил и последователност.

Приложението използва набор от шаблони за дизайн (Design Patterns), някои от които са:

* Инжектиране на зависимости / Обръщане на контрол (Dependency Injection / Inversion of control) – този шаблон се използва главно при уеб приложението, когато трябва да се инжектират класовете от бизнес слоя като услуги. Намира приложение и при тестването, когато трябва да се заменя същинските имплементации на външни методи със собствени.

Тази секция представя дизайна на решението на проблема за проекта ви. Опишете каква софтуерна платформа сте избрали за вашето решение /напр. .NET, java/. Представете схема на софтуерната архитектура на решението /по модули и/или слоеве/ с диаграма на разгръщането, както и диаграми на класовете на дизайна /с ограничения, описани на OCL/, диаграми на времето /за задаване на времена за синхронизация и комуникация в решението/ и компонентни диаграми. Илюстрирайте решението с извадки от генериран сорс код.

## Реализация на архитектурата на приложението

## Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой.

## Организация и код на заявките към база от данни

За достъп до базата данни използвам инструментариума EF Core (Entity Framework Core), който е ORM (Обектно-Релационно картографиране) библиотека за .NET платформата. EF Core позволява за работа с данните като с обекти и колекции, без да се пише SQL заявки ръчно. EF Core поддържа различни видове бази данни, като в моя проект използвам SQL Server.

За да се използва EF Core, трябва да се дефинират моделите на данните като класове в C# кода. Всеки модел има свойства, които отговарят на колоните в таблицата в базата данни.

За да se управляваt връзката с базата данни и операциите с данните, трябва да се дефинира контекст за базата данни като клас, който наследява от базовия клас DbContext. В контекстът трябва да се декларира свойства от тип DbSet<T>, където T е моделът на данните. DbSet<T> представлява колекция от обекти, които съответстват на таблица в базата данни.

За да извършваме заявки към базата данни, използваме LINQ (Language Integrated Query) синтаксис, който ни позволява да пишем заявки като изрази в C# кода. LINQ заявките се превръщат в SQL заявки от EF Core и се изпращат към базата данни. LINQ заявките могат да използват различни методи за филтриране, сортиране, групиране, проекция и агрегация на данните.

За да се добавят или изтриват обекти в базата данни, се използват методите Add, AddRange, Remove или RemoveRange на DbSet<T>. Тези методи променят само локалното състояние на обектите в контекста, без да ги променят в базата данни. За да запазим промените в базата данни, трябва да използваме метода SaveChanges или SaveChangesAsync на контекста.

## Наличие на потребителски интерфейс (конзолен, графичен, уеб)

Описание на основните функционалности на интерфейса на приложението.

Забележка: Няма формално изискване на определен брой диаграми от даден вид, за даден брой проектанти.

# Ефективност и бързодействие на решението

Съдържа описание и анализ на известните решения, като се цитират съответните литературни източници.

# Тестване

Тук се *включват тестовите случаи* и какви видове тестване предвиждате в реалното изпълнение на проекта, напр. с колко и какви документи, в какви браузъри, с какви приставки, и т.н.

# Заключение и възможно бъдещо развитие

В заключение, обобщете резултатите от работата ви по проекта, както и предимствата и ограничеността на използваните технологии / езици / методи. Укажете какви алтернативи могат да се използват и техните предимства и недостатъци. Опишете каква е използваемостта на подобни решения в практиката и какво бихте предложили като насоки за бъдещо развитие на вашето решение.

# Използвани литературни източници и Уеб сайтове

Използвайте вградената функционалност на Word: References > Citations & Bibliography

1. Уеб сайт на ….., адрес ….
2. Уеб сайт на ….., адрес ….
3. Уеб сайт на ….., адрес ….
4. Уеб сайт на ….., адрес ….
5. Литературен източник 2
6. Литературен източник 3
7. Литературен източник 4
8. Литературен източник 5

# Приложения

При необходимост можете да добавите и допълнителни секции под формата на апендикси. Таблица с диаграми, таблици и графики

Забележка:

1. Документацията на проекта се предава само в електронен вид в MS Word, чрез качването на архив с документа и останалите файлове по проекта, в задание за предаване на проект, в канала на екипа в Teams.
2. Кода на проекта, базата данни и документацията трябва да са налични в репозитори в GitHub, което е копие на заданието генерирано в организацията.