Технически университет - София
Факултет по приложна математика и информатика

Учебна дисциплина „Софтуерни технологии“

Проект: PrintMatic

Курсов проект

на

Мария Деликоцева,
Мария Магдалена Братулева,
Симеон Кюрчийски,
Борислав Каменски

Съпроводен
от

Атанас Юруков, Владимир Ганчев, Иван Дражев,
Огнян Барух, Петко Петков

София 2024

Съдържание

1. Въведение

Проектът "PrintMatic" представлява иновативна платформа, предназначена да оптимизира и автоматизира процеса на поръчки в копирни центрове. Основната цел е да се създаде леснодостъпна система, която позволява на клиентите дистанционно да качват документи, да задават предпочитания за печат и да извършват онлайн плащания. Това не само улеснява потребителите, като елиминира нуждата от физическо посещение, но и подпомага служителите, осигурявайки централизирана система за управление и проследяване на поръчките.

Този архитектурен документ има за задача да осигури детайлно представяне на системата "PrintMatic", като включва архитектурния дизайн и организационната структура на основните компоненти – клиентска и сървърна част, база данни и интеграции с външни системи. Документът е създаден да служи като ръководство за разработчиците и архитектите, участващи в поддръжката и надграждането на системата, като предлага детайлно описание на връзките и взаимодействието между отделните компоненти.

В следващите раздели ще представим структурната организация на "PrintMatic", включително ключовите архитектурни решения и технически изисквания, които гарантират ефективността и надеждността на платформата.

1.1 Участници в проекта

1.1.1 Фронтенд разработчик

Основни задачи и отговорности:

- Дизайн и реализация на потребителски интерфейс: Превръщане на дизайнерските макети в интуитивни уеб страници за клиентите, включително качване на файлове, избор на настройки за печат и преглед на статус на поръчките.
- Интеграция с бекенд: Взаимодействие с API за изпращане и получаване на данни към и от сървъра (например статус на поръчките, потвърждения за плащания и др.).

- Оптимизация: Гарантиране на бързо зареждане на страниците, за да се осигури удобство и бързина за потребителите, които правят поръчки.
- Тестове: Тестване на интерфейса за съвместимост с различни браузъри и устройства, за да се гарантира стабилността на системата.
- Технологии и инструменти: React, Tailwindcss аза стилизиране, и инструменти като Vite и npm за бързо разработване и оптимизация.

1.1.2 Бакенд разработчик

Основни задачи и отговорности:

- Създаване на API: Проектиране и разработване на API за комуникация между интерфейса и базата данни, чрез което ще се предават поръчките, данните за плащания и актуализациите на статуси.
- Логика на приложението: Разработване на функционалностите за управление на поръчките, обработка на плащания и известяване на клиентите.
- Осигуряване на сигурността: Въвеждане на мерки за защита на данните на клиентите, включително защита при онлайн плащания и сигурност на личните данни.
- Технологии и инструменти: Java с Spring Boot и Hibernate за обработка на поръчките и комуникация с базата данни.

1.1.3 Разработчик на база данни

Основни задачи и отговорности:

- Създаване и поддръжка: Проектиране на структурата на базата данни за съхраняване на поръчки, потребителски данни, информация за плащания и др.
- Оптимизация: Създаване на ефективни заявки, които да поддържат висока скорост на работа на системата при натоварване.
- Бекъп и възстановяване: Редовно архивиране на данните и стратегии за възстановяване при необходимост.
- Сигурност: Изграждане на политики за защита на данните и контрол на достъпа.
- Технологии и инструменти: MySQL за администриране и мониторинг на базата данни.

1.1.4 UI/UX дизайнер

Основни задачи и отговорности:

- Създаване на прототипи: Изработка на макети на интерфейса, за да се визуализира процесът на поръчка, плащане и проследяване на поръчките.
- Дизайн на интерфейс: Определяне на стиловите елементи и изграждане на лесен за използване интерфейс за потребителите на PrintMatic.
- Изготвяне на потребителски пътеки: Проектиране на потребителски сценарии, които да осигурят гладък процес на поръчка.
- Тестване на дизайна: Извършване на тестове и корекции за оптимизиране на удобството на потребителите.
- Технологии и инструменти: Figma за създаване на визуални прототипи и взаимодействие със заинтересованите страни.

1.1.5 Софтуерен архитект

Основни задачи и отговорности:

- Дефиниране на структурата: Определяне на основните компоненти на системата PrintMatic и взаимовръзките между тях (клиент-сървър-база данни).
- Техническа стратегия: Избор на подходящи технологии и платформи за дългосрочната поддръжка на PrintMatic, като се съобразят с очакваното натоварване и сигурност.
- Оптимизация: Проектиране на системата с възможност за мащабиране при увеличаване на броя потребители.
- Сигурност: Приложение на добри практики за защита на чувствителни данни и онлайн транзакции.
- Сътрудничество: Съгласуване на архитектурния дизайн с фронтенд и бекенд разработчиците, както и с UI/UX дизайнера за хармонизиране на бизнес целите.

1.1.6 QA

Основни задачи и отговорности:

- Планиране на тестове: Създаване на тестови сценарии на базата на функционалните изисквания на PrintMatic.

- Ръчно тестване: Изпълнение на ръчни тестове за откриване на грешки в интерфейса и функционалността на системата.
- Автоматизация на тестове: Изграждане на автоматизирани тестове, които да гарантират стабилността на PrintMatic при актуализации и разширения.
- Документация: Регистриране на проблемите и сътрудничество с разработчиците за своевременно разрешаване.
- Сътрудничество: Работа с целия екип за поддържане на качеството на проекта и гарантиране на позитивно потребителско изживяване.

2. Предназначение

2.1 Обхват

2.1.1. Планиране

- Оценка на текущите нужди и изисквания: Определяне на основните функционалности за обработка на поръчки, онлайн плащания, управление на профили и комуникация между клиентите и служителите.
- Определяне на ресурсите: Изискване на екип от фронтенд и бекенд разработчици, UI/UX дизайнери и QA специалисти за разработката и поддръжката на системата.
- Закрепване на времеви рамки: Определяне на срокове за всяка фаза на проекта – от анализ на изискванията до тестване и внедряване.

2.1.2. Анализ на изискванията

- Проучване на потребителски сценарии: Определяне на основните сценарии на използване, включително процеса на подаване на поръчка за печат, следене на статуси и управление на клиентския баланс.
- Определяне на основните функционалности: Създаване на интерфейси за качване на документи, настройки на поръчки, известия за статуса и онлайн плащания.

2.1.3. Дизайн

- Архитектурно планиране: Определяне на софтуерната архитектура и базата данни, както и избор на технологии като React за фронтенд, Spring Boot за бекенд и MySQL за базата данни.
- UI/UX дизайн: Създаване на wireframes и прототипи за основните екрани на приложението, включително процеса на поръчка и управление на профила, както и определяне на цветова палитра и стил.
- Планиране на инфраструктурата: Избор на хостинг платформа и изчисляване на нуждите от мащабируемост за оптимално функциониране на системата.

2.1.4. Програмиране

- Фронтенд: Разработка на потребителския интерфейс с интеграция към бекенда за обработка на поръчки, статуси и плащания.
- Бекенд: Създаване на API-та за комуникация между фронтенд и бекенд, както и логика за управление на поръчките, плащанията и известията.
- База данни: Оптимизиране на структурата на базата данни за съхранение на потребителски и поръчкови данни, осигуряване на сигурността и целостта на информацията.

2.1.5. Тестване

- Функционални тестове: Проверка дали всички функционалности, включително качване на документи, настройка на поръчки и плащания, работят според изискванията.
- Тестване на производителността: Тестване на приложението при различни обеми на трафика, за да се гарантира стабилност и скорост.
- Сигурност: Проверка на уязвимостите на системата, особено за защита на потребителските данни и сигурността на онлайн плащанията.

2.1.6. Деплоймънт

- Избор на платформа за разпространение: Внедряване на системата в подходяща cloud платформа за лесно управление и мащабиране.
- Оптимизация за продуктивна среда: Подобряване на скоростта и ефективността на приложението, за да се осигури плавна работа за клиентите.

- Мониторинг след пускане: Непрекъснат мониторинг на системата за проследяване на производителността и разрешаване на възникнали проблеми след пускането в експлоатация.

3. Актьори

3.1. Клиенти

- Поръчки за печат: Възможност за създаване на нова поръчка чрез качване на документи, избор на параметри за печат, като брой копия, цветност и формат.
- Профил и баланс: Управление на личния профил, включително настройка на информация и наблюдение на наличния баланс.
- Следене на статус: Получаване на информация за статуса на поръчката, като „В процес," „Изпълнена," или „Отказана."
- Система за известия: Известяване чрез имейл или в приложението при завършване на поръчка или промяна на статуса на поръчката.
- Онлайн плащания: Възможност за извършване на плащания за поръчките директно в приложението чрез добавяне на средства към акаунта или плащане на индивидуална поръчка.

3.2. Служители на копирния център

- Административен панел за поръчки: Достъп до панел за управление, където могат да преглеждат всички активни поръчки и техните спецификации, както и да актуализират статуса на поръчките.
- Обработка на поръчки: Възможност за промяна на статусите на поръчките според прогреса им, като „В процес на изпълнение" или „Изпълнена."

3.3. Администратори

- Административен панел: Управление на потребители, създаване и премахване на акаунти за клиенти и служители.

- Панел с предлаганите услуги и възможност цената им да бъде редактирана от администратор.

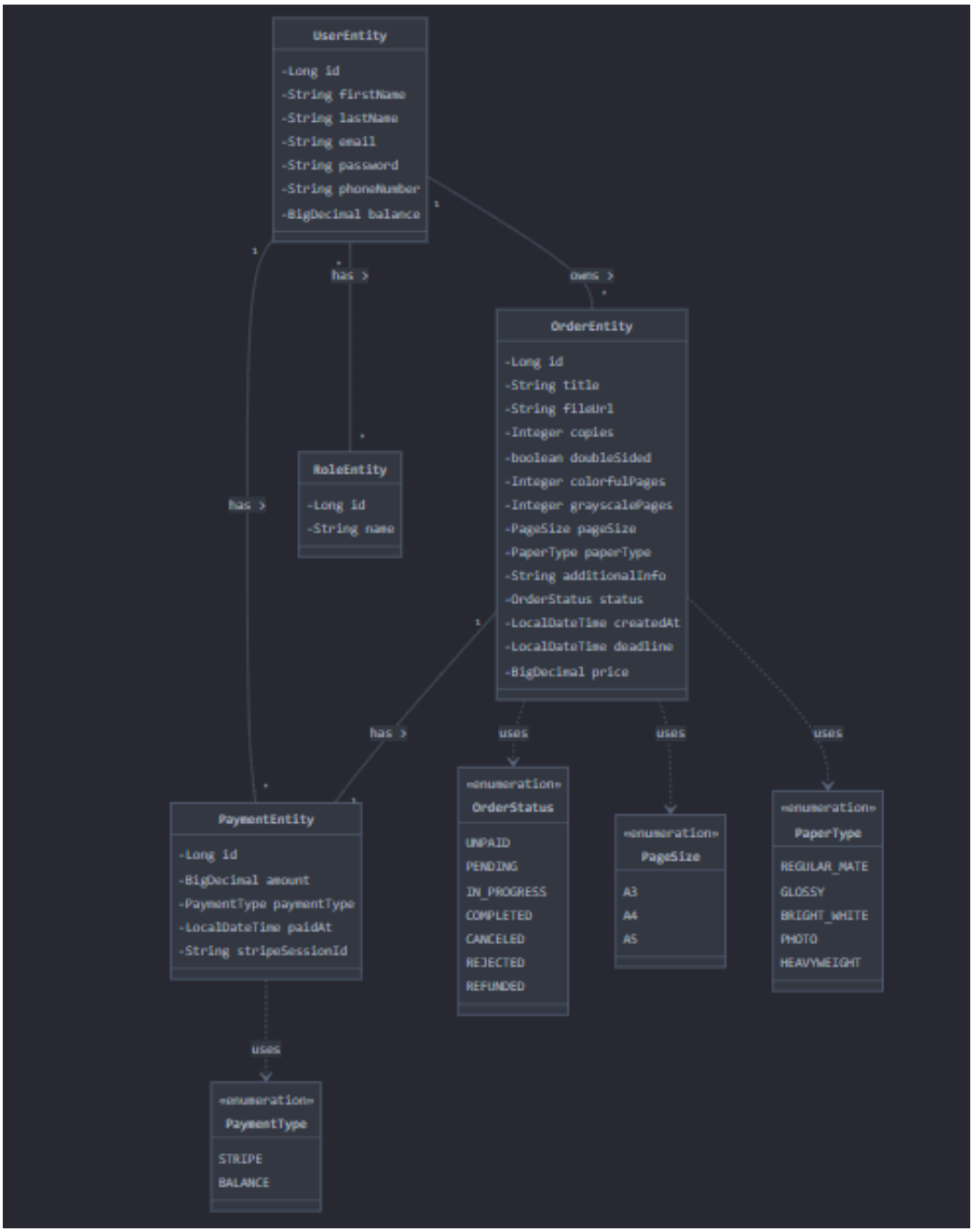
4. Архитектурен обзор

4.1 Use-case изглед

Описание на функционалността:

- **Регистрация на потребител:**
    - **Актьор:** Потребител
    - **Сценарий:** Потребителят попълва форма с имейл, парола и име. Системата валидира данните и създава нов профил. След успешно регистриране, потребителят се прехвърля към началната страница в профила си.
- **Вход в системата:**
    - **Актьор:** Потребител
    - **Сценарий:** Потребителят въвежда потребителско име и парола. Системата проверява данните и допуска потребителя в профила.
- **Извършване на плащане:**
    - **Актьор:** Потребител
    - **Сценарий:** Потребителят качва файла с документа за принтиране, въвежда желаните характеристики (размер и тип на хартията, брой копия и др.) и избира метод на плащане чрез баланс или непосредствено заплащане на поръчката чрез Stripe . След потвърждаване, системата обработва плащането и актуализира статуса на поръчката.
- **Преглед на поръчки:**
    - **Актьор:** Потребител
    - **Сценарий:** Потребителят има достъп до историята на поръчките си, включително статус им.

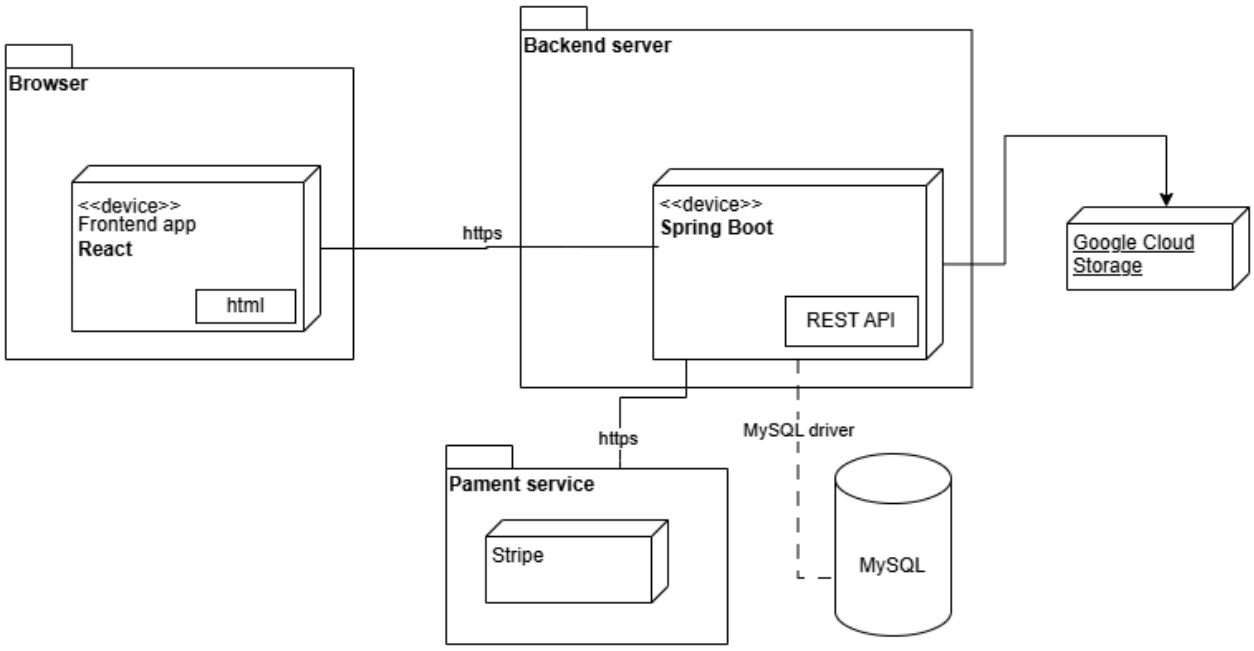
4.2 Логически изглед



4.3 Изглед на внедрянавето

Изгледът на внедряване представя топологията на софтуерните компоненти на физическия слой, физическите връзки между тези компоненти. Описва

разполагането на софтуерните компоненти върху хардуера и показва разпределените аспекти на системата. Дава ни обобщена представа за инфраструктурата и гарантира стабилността и скалируемата на системата.



На предоставената deployment диаграма са изобразени различните компоненти и взаимовръзките между тях, които формират архитектурата на web приложението. Ето детайлно обяснение на всеки компонент:

Browser:

Софтуерът е web application и се достъпва посредством web browser.

Frontend application React: Това е клиентската част на приложението, която е разработена с помощта на библиотеката React. React се използва за създаване на интерактивни потребителски интерфейси (UI). С нейна помощ в проекта ще се реализират UI за всяка от ролите на потребителите на системата (admin, client, operator). Комуникира със сървъра чрез HTTPS протокол.

Backend server:

Spring Boot: Java framework, който предоставя анотации, чрез които се улеснява създаването на REST API. Дава възможност за достъпна и опростена интеграция с бази данни и други външни услуги. В него ще се представи основният модел на данните, работата с тях, и ще се реализира бизнес логиката на приложението.

REST API:

REST API позволява на клиентите да взаимодействат със сървъра чрез HTTP заявки. Тази интерфейсна част улеснява обмена на данни между клиент и сървър.

MySQL:

MySQL Driver: Това е софтуерната част, която улеснява комуникацията между backend сървъра и MySQL базата данни. Драйвърът интерпретира заявките на API-то и ги превръща в заявки, които базата данни може да обработва.

MySQL Database: Релационна база данни, която съхранява данните на системата, включително потребителска информация, данни за поръчките и транзакциите.

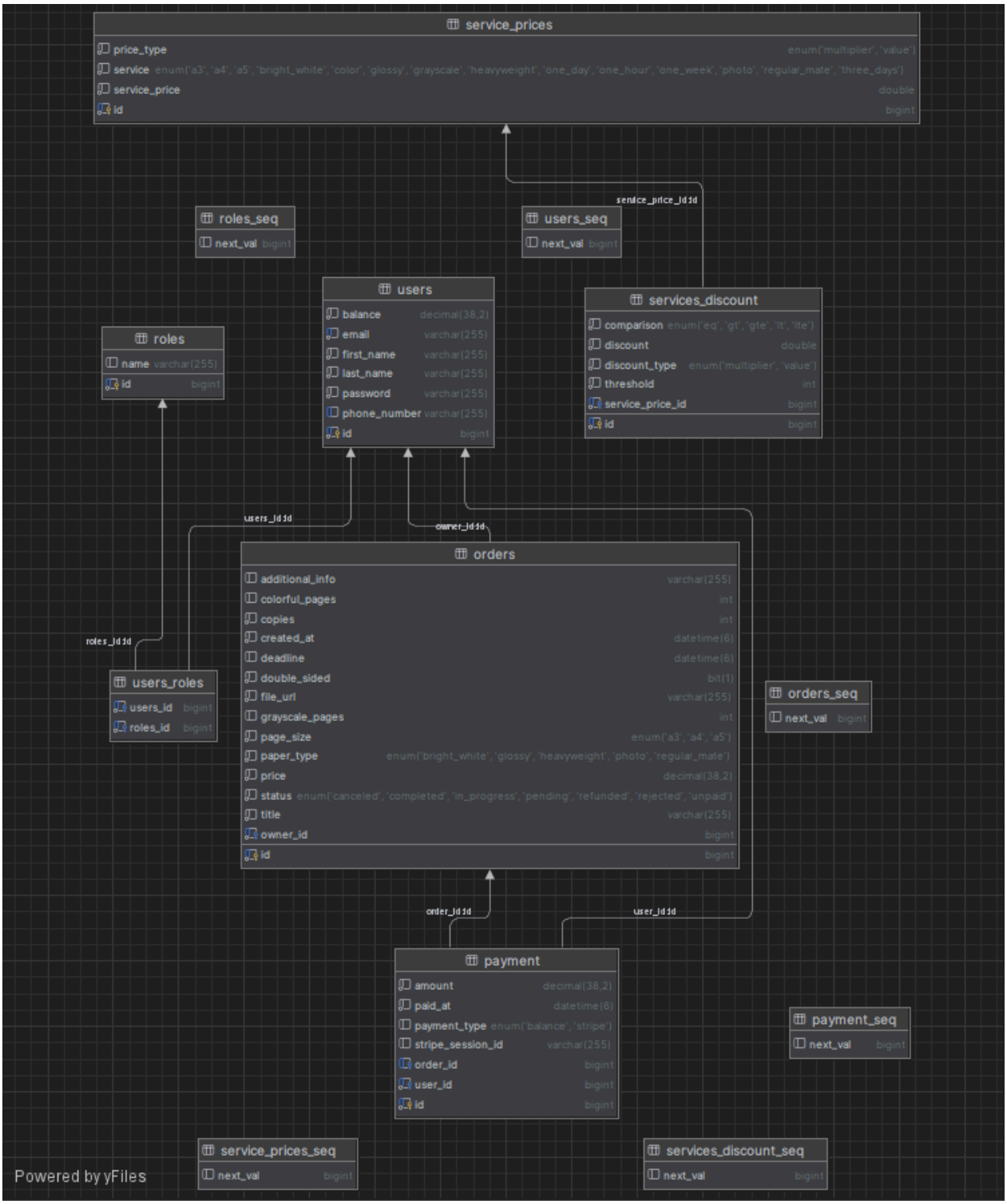
Payment Service: Платформа за обработка на плащания, която улеснява транзакциите между клиентите и търговците. Ще гарантира сигурен и ефективен начин за плащане при поръчка, както за зареждане на баланс, така и при единично плащане за поръчки.

Google cloud storage: Облачна услуга за управление на файлове. Основната услуга за съхранение на файловете, които клиентите качват за печат в копирния център. Само копирният център или авторизиран персонал ще имат достъп до качените файлове за печат.

Диаграмата показва как клиентската и сървърната части на приложението са взаимосвързани и как трафикът между тях е управляван за осигуряване на бързина, сигурност и мащабируемост.

4.4 Изглед на данните

Изгледът да данните дава по-ясна представа за модела на данните. Дава възможност за по-нататъшно развитие, направа на промени и добавяне на нови функционалности с оглед на съществуващата структура.



Users: Представя всички потребители на приложението. Предоставя информация за имената на лицето, паролата на неговия акаунт, имейл адреса (уникален и с предназначение като потребителско име), баланс. Потребителите се асоциират с

различни роли в системата и притежават различни права. Чрез връзката много към много с ролите се дава възможност на операторите в копирния център, също да са в ролята на клиенти и да се достъпват до услугите на копирния център.

Roles: Основните роди са на клиента, оператора в копирния център и админа. Те дават ясна представа за това до какво може да достъпва съответният потребител, неговите права и характеристики.

Orders: Представят поръчките, който клиентите изпращат към операторите в копирния център. Включва информация за файла с документа за разпечатване, както и основни характеристики обвързани с него – броят на цветните и на черно белите страници, размерът(A3, A4, A5 и т.н.) и типът на желаната хартия (обикновена матирана, гланцова, ярка бяла, фото, тежка или още позната като кадастрон и т.н.), заедно с друга важна информация като цена и краен срок за изпълнение. Поръчките са обвързани с клиентите, които ги правят като един клиент може да прави множество поръчки (връзка едно към много).

Payment: Тя съдържа основна информация при извършване на разплащанията, като датата на извършената транзакция, типа ѝ и цената. Притежава връзка, както с клиента направил поръчката, така и с поръчката, към която се отнася.

ServicePrice: Тя съдържа информация за всички предоставяни услуги, вида на цената (базова цена или надценка), и цената.

4.5 Изглед на имплементация

4.5.1 Концепция на слоеве:

Потребителски интерфейс (UI Layer)
Отнася се до всички компоненти, с които потребителите на системата взаимодействат. Този слой обхваща уеб страниците и другите интерфейсни елементи, като те трябва да са приспособени и изградени с цел удобство и яснота при използване. Обикновено включва елементи като визуални компоненти (бутони, форми, текстови полета) и логика за представяне, която преобразува данните от backend-а във вид, който потребителят може лесно да разбере.

Слой на контролери (Controller Layer)
Свръзката между потребителският интерфейс и логиката на приложението. В този слой се обработват заявките настъпили от потребителите, чрез по-нататъшното извикване на сервизите и изпълнението на бизнес логиката зад тях се връща отговор обратно към потребителският интерфейс. Те

следят за правилното изпълнение на заявките и връщат резултати или съобщения за грешки.

Слой на сервизите (Service Layer)
Това е слой за бизнес логиката на приложението. Той обработва данните, изпълнява логически операции и координира достъпа до данни. Чрез него приложението извършва своите основни функционалности като ги разделя на отделни, независими модули, които на свои ред изпълняват конкретни задачи. Притежава пряка връзка с repository layer – а за достъпване и съхранение на данните в базата.

Слой на репозиторитата (Repository Layer)
Този слой е пряката комуникация с базата данни. Грижи се за извършването на основните CRUD операции, които включват създаването на нов запис, неговото четене, изтриване и актуализиране. Слоят е отговорен за съхранението и управлението на информацията.

Слой на сигурността (Security Layer)
Отговорен автентикацията и оторизацията, за да осигури сигурност на данните. Контрола на достъпа, който този слой гарантира е пряко свързан с ролите на потребителите в системата.

4.5.2 Цел на слоевете:

Разделение на отговорностите

Използването на слоеве с конкретни отговорности и роли прави системата по-структурирана и нейното изработване и управление по-ефективно. Модулността гарантира, че слоевете са независими един от друг, което подпомага за постигането на по-лесна промяна в съществуващата логика и дори нейното надграждане.

Тестване и поддръжка

Разделянето на слоеве позволява по-лесно тестване на отделни части от системата, тъй като всеки слой може да се тества самостоятелно чрез unit или integration тестове.

Съхранение на конфиденциална информация

Информацията, която трябва да бъде съхранена и за нея да се осигури дадено нова на сигурност, може да се пази на отделен слой, който предоставя ограничен достъп.

4.5.3 Правилата за тяхната реализация и употреба:

При потребителския интерфейс (UI Layer)

Той трябва да предоставя консистентност, като използва разработени стандартни UI компоненти, които да са реализирани с идеята да бъдат ясни и удобни за крайният потребител .Осъществява връзка с контролерите и да не осъществява директна връзка с слоевете отговорни за логиката и връзката с базата (сервизите и репозиторитата).

При слоя на контролерите (Controller Layer)

Да предоставя връзката между потребителският интерфейс и сервизите, без да има в себе си бизнес логика. Тази логика трябва да се изпълнява от методите намиращи се в сервизите като контролерите са съсредоточават изцяло над обработката на заявките и връщането на информация обратно към UI слоя.

При слоя на сервизите (Service Layer)

Да се гарантира целостта на данните и изпълнението на бизнес логиката като не се осъществява директна връзка с базата.

При слоя на репозиторитата (Repository Layer)

Да отговаря за основните CRUD операции, осъществявайки връзката с базата. Не бива да съдържа бизнес логика.

Тестване и валидация

Всеки от слоевете трябва да бъде тестван за правилното си функциониране. Използват се методи за автоматично тестване, за да се провери коректността на работата и отговорностите на отделният слой.

7. Нефункционални изисквания

7.1 Надеждност и възстановимост :

За поддържане на достъпността на системата, копирният център ще използва архитектура, която поддържа резервни копия на данните и автоматични бекъпи, за да се предпази от загуба на информация при неуспешни операции.

Тактики за откриване на проблеми: Наблюдение на работните процеси чрез логиране на важни действия и изпращане на известия при откриване на неочаквани събития или грешки.

Тактики за възстановяване: Поддръжка на резервни копия на ключовите данни (например статуси на поръчките), с възможност за възстановяване в случай на загуба на данни.

7.2 Разширяемост:

Възможности за бъдещи промени в софтуера:

Тактики за разширяемост: Прилагане на архитектурен шаблон MVC (Model-View-Controller), който позволява нови модули и услуги (напр. нови видове плащания) да се интегрират лесно.

Технологии: Използване на стандартизирани RESTful API за връзка между основните модули и комуникация с външни системи.

Дизайнерски похвати: Използване на обектно-ориентирани принципи, като инкапсулация и наследяване, за лесно добавяне на нови функционалности.

7.3 Сигурност:

Тактики за ограничаване на достъпа: Осигуряване на различни роли с права на достъп, като клиент, служител и администратор.

7.4 Интероперабилност:

Анализ на входно-изходните канали за обмен на информация:

Системата ще поддържа интеграция с различни външни системи (като платежни услуги) чрез стандартизирани API комуникации.

Тактики за поддръжка на различни интерфейси: Поддържане на JSON/XML формати за лесна интеграция с различни платформи и поддръжка на различни формати за документи, напр. PDF

7.5 Използваемост:

Интерфейс за комуникация между потребител и софтуер:

Потребителският интерфейс ще е изчистен и интуитивен, като всички основни функционалности ще бъдат достъпни само с няколко клика.

Обратна връзка: Интегриране на известия и имейли за промяна на статусите на поръчките, за да се информират клиентите навременно относно техните заявки.