Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работная работа по предмету «Проектирование информационных систем» на тему «Объектно-ориентированное модулирование. Физические диаграммы UML»

Студент: Ковкель Н. В.

ФИТ 4 курс 4 группа

Преподаватель: Якубенко К. Д.

Минск 2024

# 1 Постановка задач

Задачей проекта является моделирование процессов информационной системы на основе методологии IDEF3, разработка моделей бизнес–процессов и их декомпозиции. Модели должны отражать все функциональные требования, заявленные к информационной системе на основании предыдущих лабораторных работ.

Web–приложение должно быть выполнено с использованием асинхронного программирования, взаимодействовать с базой данных, реализовано под разными платформами. Web–приложение должно представлять собой web–приложение с асинхронным UI с использованием фреймворка React Native. Отображение, бизнес–логика и хранилище данных должны быть максимально независимы друг от друга для возможности расширения. Диаграмму вариантов использования разработать на основе UML, также необходимо разработать логическую схему базы данных и структурную схему приложения. Язык разработки проекта Python, платформа «Django Rest Framework».

Функционально web-приложение должно:

* поддерживать роли гостя, пользователя;

Обеспечивать гостям возможности:

* зарегистрироваться;
* аутентифицироваться;
* авторизоваться.

Обеспечивать пользователям возможности:

* взаимодействовать через свайп-систему с потоком людей;
* ставить лайки;
* пропускать анкеты;
* редактировать профиль (логин, пароль, аватар, предпочтения);
* получать уведомления о новых сообщениях и взаимных симпатиях;
* общаться с другими пользователями через чат.

Основные страницы веб-приложения:

* страница регистрации;
* страница авторизации;
* главная страница;
* личный кабинет пользователя;
* страница уведомлений;
* чат;
* настройки;
* страница помощи и поддержки.

Для гарантированной безопасности пользователей приложение должно применять метод хеширования паролей SHA256 перед их сохранением в базу данных.

# 2 Описание программных средств

Draw.io является мощным инструментом для создания диаграмм и визуального моделирования, который использовался для разработки и документирования архитектуры системы в процессе выполнения лабораторной работы. Приложение предоставило интуитивно понятный графический интерфейс для создания структурных схем, которые помогли визуализировать ключевые компоненты системы, а также их взаимосвязи и процессы взаимодействия. Использование draw.io стало важным шагом для проектирования системы, так как оно упростило процесс создания диаграмм, описывающих функциональные блоки приложения и их взаимодействие. Приложение также поддерживает интеграцию с различными облачными сервисами, что облегчило совместную работу над схемами и предоставило гибкость в управлении проектами.

Инструмент draw.io был выбран за его следующие особенности:

* Доступность: кроссплатформенная, поддержка работы в браузере, а также на Windows, macOS и Linux.
* Поддержка стандартов: возможность создания UML–диаграмм, блок–схем, диаграмм ER, IDEF0 и IDEF3, что позволило эффективно визуализировать бизнес–процессы и архитектуру системы.
* Легкость в использовании: интуитивно понятный интерфейс, который позволяет создавать диаграммы без необходимости владения сложными графическими навыками.
* Интеграция с облачными хранилищами: поддержка Google Drive, OneDrive, GitHub, что упрощает доступ к схемам и их совместное редактирование.

Основное назначение draw.io в этом проекте заключалось в создании визуальных моделей бизнес–процессов и архитектурных решений, которые легли в основу разработки системы. Эти диаграммы помогли в структурировании процессов разработки и визуализации потоков данных, что обеспечило более четкое понимание системы в целом.

Данные о технологии draw.io:

* Разработчик: JGraph Ltd.
* Адрес загрузки: [app.diagrams.net](https://app.diagrams.net).
* Использовался для: создания диаграмм и визуальных моделей бизнес–процессов и архитектурных решений системы.
* Доступность: кроссплатформенная, поддержка веб–версии и настольных приложений.

# 3 Описание практического задания

Приложение «EventFlow» будет выполнено в монолитной архитектуре, где все компоненты системы будут интегрированы в единое приложение, что обеспечит централизованное управление и взаимодействие между функциональными модулями. Основные компоненты системы включают:

* PostgreSQL Server.
* User Management Module.
* Messages Module.
* Chat Management Module.
* React Native Client Application

Весь сервер будет развернут на Docker.

Сервис клиента будет использовать в браузере на ОС Windows 11. Он будет использовать библиотеку React Native, для хранения будет использовать Redux ToolKit, для стилизации сайта будет использоваться библиотека Bootstrap.

Для стилизации интерфейса будет использоваться библиотека Bootstrap, которая предлагает широкий набор готовых компонентов и утилит. Это не только ускорит процесс разработки, но и обеспечит адаптивный дизайн, что важно для пользователей, работающих на различных устройствах. Bootstrap позволит легко настраивать и стилизовать элементы интерфейса, сохраняя при этом единый стиль приложения.

На серверной стороне будет разработан основной сервис с использованием Python и фреймворка Django Rest Framework. Это обеспечит высокую производительность и безопасность, а также упростит создание API для взаимодействия с клиентским приложением. Django Rest Framework предоставит мощные инструменты для сериализации данных и аутентификации пользователей, что значительно ускорит процесс разработки и поддержки.

Все сервисы будут развернуты в контейнерах Docker, что позволит создать изолированную среду для каждого компонента системы. Это обеспечит гибкость в развертывании и масштабировании приложения, а также упростит процесс тестирования и отладки. Использование единой базы данных для хранения всех данных обеспечит целостность и согласованность данных, что критично для надежности системы. Так же Docker повышает безопасность приложения, за счет запуска в контейнере где пользователь, не будет иметь возможность взаимодействовать с файловой структурой контейнера.

Работа с базой данных будет осуществляться через Django ORM, что упростит взаимодействие с различными типами баз данных и позволит разработчикам сосредоточиться на бизнес-логике приложения, а не на низкоуровневых деталях SQL-запросов. Django ORM предоставляет удобный и понятный интерфейс для выполнения запросов, что значительно ускорит процесс разработки и улучшит поддержку кода в будущем, а так же позволит в случае необходимости быстро изменить структур базы данных.

Таким образом, выбранные технологии и архитектура обеспечат создание высококачественного, масштабируемого и удобного в обслуживании приложения, способного эффективно справляться с требованиями пользователей и бизнес-процессами.

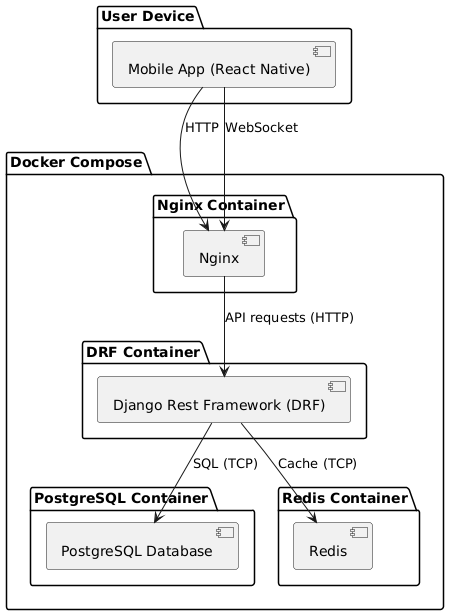
=

Рисунок 3.1 – Диаграмма развертывания

На диаграмме видны все описанные компоненты приложения.

Компонента Redis Server – это In-Memory DB, которая будет отвечать за хранения токенов авторизации или кодов для авторизации. Данный сервер будет развернут на докере на версии 7.4.0. Обращение к этой компоненте будет происходить по протоколу TCP.

Компонента PostgreSQL Server – это СУБД, которая будет отвечать за хранения всей информации о состоянии работы системы и с системой. Обращение к ней будет по протоколу TCP.

# 4 Ответы на вопросы

1. Укажите назначение физических диаграмм: компонентов и развертывания.

Диаграмма компонентов позволяет определить состав программных компонентов в роли которых может выступать исходный, бинарный и исполняемый код.

Диаграмма развёртывания же позволяет определить распределение компонентов системы по её физическим узлам; показать физические связи между всеми узлами реализации системы на этапе ее выполнения; выявить узкие места системы и реконфигурировать её топологию для достижения требуемой производительности.

2. Дайте определение понятиям: узел, артефакт, интерфейс.

Артефакт – некая физическая сущность, программный компонент, который используется или создаётся во время работы программного обеспечения.

Узел – то, что может содержать программное обеспечение (некоторый физически существующий элемент системы, обладающий вычислительным ресурсом).

Интерфейс – элемент, который служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры.

3. Опишите нотации, которые используются для представления компонентов.

Для представления компонентов используются две нотации – рисунок 2, отличаются они визуально.

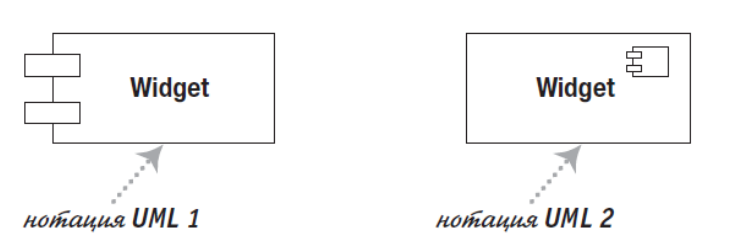


Рисунок 2 – Нотации для представления компонентов

4. Опишите основные нотации, которые используются для представления архитектуры системы в виде диаграммы развертывания.

Узлы (Nodes): Представляют физические устройства или вычислительные ресурсы, такие как серверы, ПК, мобильные устройства и т.д. Узлы изображаются в виде прямоугольников с именем устройства.

Артефакты (Artifacts): Представляют программные компоненты или данные, которые размещены на узлах. Они изображаются в виде прямоугольников или эллипсов внутри узлов.

Связи (Connections): Показывают взаимодействие между узлами. Они могут быть направленными или двунаправленными линиями, обозначающими поток данных, коммуникации или зависимостей между узлами.

5. Укажите основные виды связей между компонентами и между узлами

Сетевые связи (Network Links): Показывают физические сетевые соединения между узлами. Это могут быть Ethernet, Wi-Fi, VPN и другие типы сетевых соединений.

Коммуникационные связи (Communication Links): Представляют логические связи между компонентами системы. Это могут быть протоколы обмена данными, такие как HTTP, TCP/IP, MQTT и другие, которые описывают способы обмена информацией между компонентами.

Зависимости (Dependencies): Показывают, какие компоненты или узлы зависят от других. Например, если один компонент зависит от базы данных, это может быть представлено стрелкой или связью, указывающей на эту зависимость.

Репликация и балансировка нагрузки (Replication and Load Balancing): Если компоненты или данные реплицируются на несколько узлов или происходит балансировка нагрузки между узлами для повышения производительности и надежности, это может быть показано на диаграмме развертывания.

Прокси и шлюзы (Proxies and Gateways): Представляют промежуточные узлы, используемые для обеспечения безопасности, переадресации или трансляции данных между компонентами.