



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ: ИУ Информатика, искусственный интеллект и системы управления

КАФЕДРА: ИУЗ Информационные системы и телекоммуникации

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ НА ТЕМУ:

Информационная система фитнес-центра

Студент **ИУЗ-53Б**
(Группа)

(Подпись, дата)

Протопопова В.О.
(И.О.Фамилия)

Руководитель
(Подпись, дата)

(И.О.Фамилия)

Недашковский В.М.

Консультант
(Подпись, дата)

(И.О.Фамилия)

Недашковский В.М.

2024 г.

Оглавление

ВВЕДЕНИЕ	3
Описание предметной области	5
Функциональная модель бизнес процессов	7
Глоссарий предметной области.....	10
Видение	12
Концептуальная модель предметной области.....	15
Функциональные требования	16
Диаграммы вариантов использования.....	19
Краткое и подробное описание вариантов использования.....	20
Диаграммы анализа	29
Диаграмма последовательности	30
Диаграммы классов этапа проектирования	32
Схема (структура) базы данных	34
Требования к функциональным характеристикам.....	35
Требования к надежности.....	38
Условия эксплуатации	39
Требования к составу и параметрам технических средств	40
Требования к информационной и программной совместимости	42
Требования к программной документации	42
Технико–экономические показатели	43
Стадии и этапы разработки.....	44
Архитектура системы	45
База данных.....	49
Серверная часть системы.....	53
Интерфейс пользователя	59
Анализ исходных данных.....	66
Тестирующие драйверы и заглушки	66
Протокол тестирования.....	68
Исследование системы на производительность и отказоустойчивость	69
ЗАКЛЮЧЕНИЕ.....	71
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	72

ВВЕДЕНИЕ

Для выполнения данного задания будет создана информационная система для фитнес-центра.

В современном мире фитнес-центры играют важную роль в обеспечении здоровья и физической активности для многих людей. Однако, чтобы эффективно управлять фитнес-центром и предоставлять качественные услуги, им требуется надежная и эффективная информационная система.

Информационная система фитнес-центра является ключевым инструментом, позволяющим управлять клиентской базой, мониторить состояние оборудования, проводить анализ данных и предоставлять новые тренировочные программы. Управление данными в информационной системе фитнес-центра имеет стратегическое значение. Фитнес-центры собирают большое количество данных, таких как информация о клиентах, их тренировках, питании и здоровье. Эффективное управление этими данными позволяет фитнес-центрам оптимизировать тренировочные процессы, адаптировать программы под индивидуальные потребности клиентов, выявлять тренды в фитнес-индустрии и обеспечивать высокое качество обслуживания.

Прежде всего нужно словесно описать предметную область. Затем необходимо составить глоссарий предметной области, где будут расписаны основные термины и понятия. После этого нужно составить личное видение этого проекта, а также составить концептуальную модель. Для успешной разработки информационной системы важно создать диаграммы вариантов использования, которые помогут визуализировать функциональные требования к системе. Только после того, как эти требования будут четко определены и отображены на диаграммах, можно перейти к следующему этапу проектирования, а именно созданию диаграммы классов.

Диаграмма классов представляет собой визуальную модель, которая описывает структуру объектов в системе и их взаимодействие. Она строится на основе сформулированных вариантов использования и представляет собой набор классов, отражающих сущности и их свойства в системе. После построения диаграммы классов можно перейти к разработке структуры базы данных, используя все классы этапа проектирования.

Первый раздел

Этап подготовки

Описание предметной области

Клиент, желающий воспользоваться **услугами** фитнес-центра, может обратиться в офис лично или воспользоваться онлайн-ресурсами. Там его приветствует сотрудник (менеджер), который проводит консультацию с клиентом с целью определения его фитнес-цели, предпочтений и требований. На основе полученной информации **менеджер** составляет **карту клиента**, учёт которой ведётся в **списке клиентов**. Для каждого клиента ведётся персональная учетная запись, в которой хранятся его персональные данные: ФИО, возраст, контактные данные.

Менеджер фитнес-центра предлагает различные виды **членства**, информация о которых хранится в **каталоге членств**; учёт всех заявок на членства ведётся в **журнале заявок**. Сотрудник по необходимости рассказывает о доступных услугах и помогает клиенту выбрать наиболее подходящий вариант. Каждый вид членства имеет свои уникальные характеристики, такие как название, стоимость, количество посещений, срок действия и перечень предоставляемых услуг. Например, одно членство может давать доступ только к групповым тренировкам, а другое имеет доступ так же к персональному тренеру. Кроме того, у каждого членства может быть свой набор условий, таких как возможность заморозки на время отсутствия клиента.

В фитнес-центре работают **тренеры**, которые проводят индивидуальные и групповые тренировки с клиентами. Для каждого тренера ведётся **карта тренера**. В карте хранятся данные, включая ФИО, возраст, стаж и рейтинг. Ведётся **список тренеров**.

Тренер также может оценить физическую форму клиента и предложить рекомендации по выбору подходящих **тренировок и программ**. Ведётся

список тренировок и программ. Клиент также имеет возможность выбрать тренера из числа доступных в фитнес-центре. Менеджер может предоставить информацию о квалификации и специализации каждого тренера, а также их расписание доступности. Это позволяет клиенту выбрать тренера, который наилучшим образом соответствует его целям и предпочтениям.

После выбора плана членства, программ тренировок и тренера, менеджер оформляет заказ и регистрирует клиента в информационной системе фитнес-центра. Это включает указание выбранного членства, программ тренировок и выбранного тренера. Также менеджер может создать уникальный личный аккаунт для клиента, где будут отображаться его тренировки, достижения и прогресс, а также информация о тренере, назначенном ему на занятия.

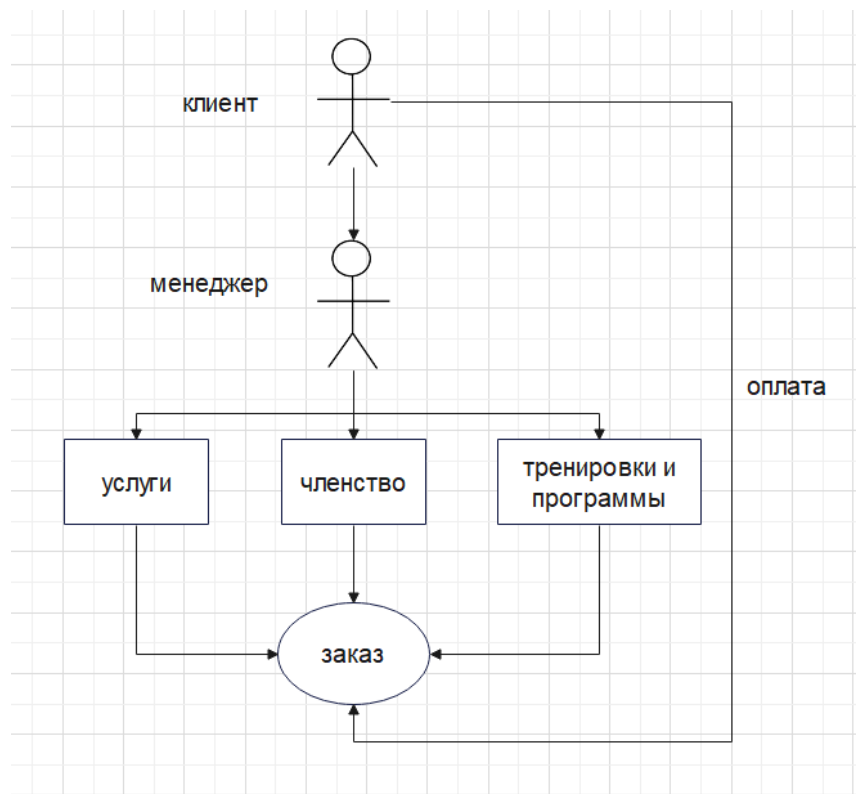


Рисунок 1 - Предметная область

Функциональная модель бизнес процессов

Модель бизнес-процессов будут представлены в виде ICOM (Input-Control-Output-Mechanism) модели, лежащей в основе нотации IDEF0 (Integration DEfinition for Function Modelling).

Контекстная диаграмма бизнес-процесса для фитнес-центра:

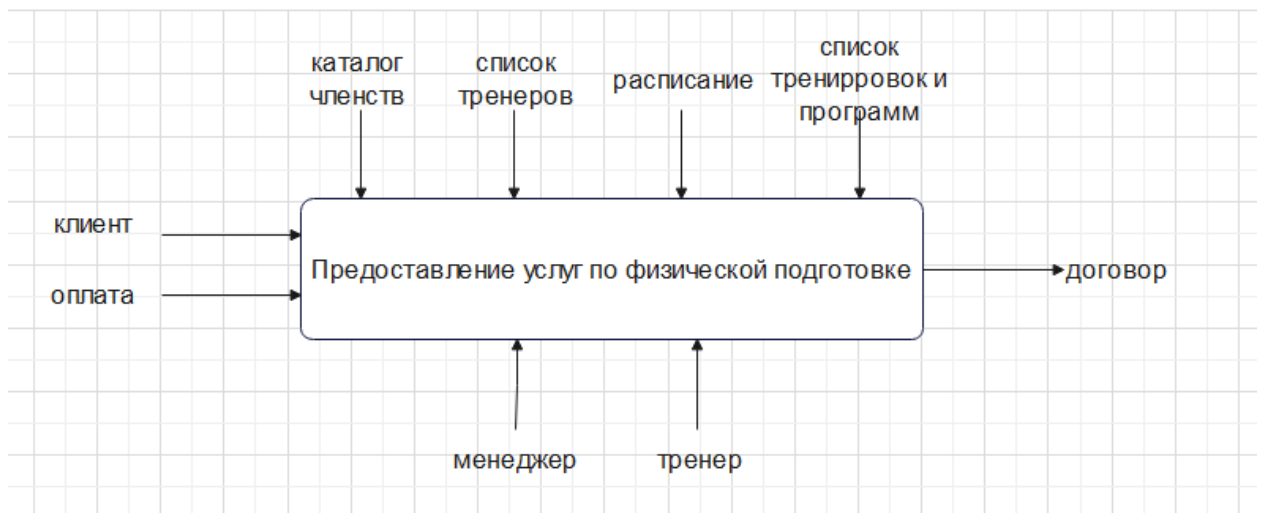


Рисунок 2 - Контекстная диаграмма БП

Входы – т.е. информация, подвергаемая обработке:

- «Оплата»
- «Клиент»

Выходы – т.е. данные, полученные в результате выполнения работы:

- «Договор» - документ с прописанными обязательствами клиента и компании.

Управление – условия, ограничения, указания:

- «Каталог членств»
- «Список тренеров»
- «Расписание»

- «Список тренировок и программ»

Механизмы или исполнители – те, кто выполняет работу:

- «Менеджер»
- «Тренер»

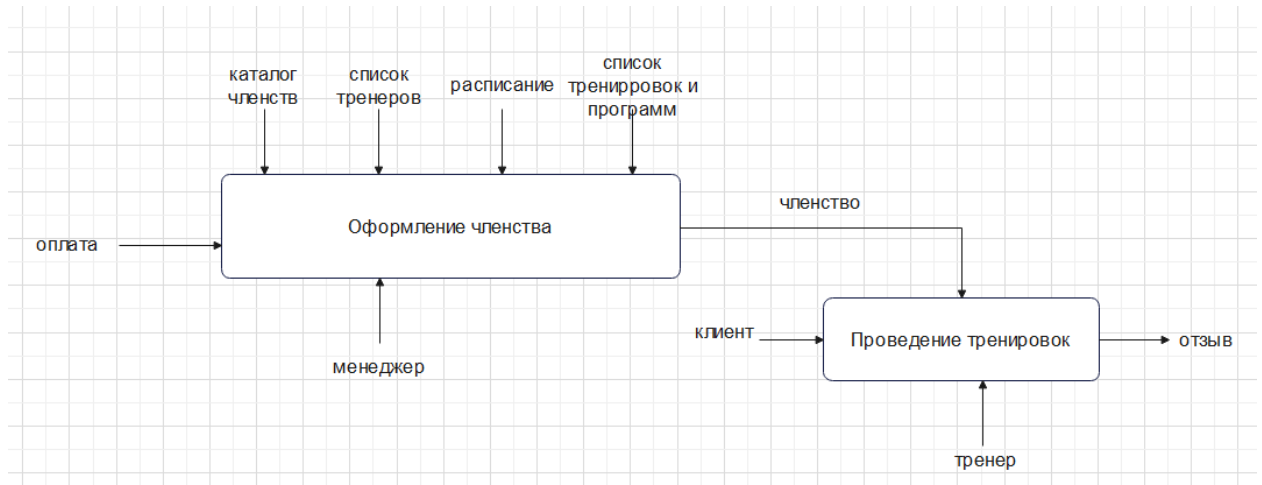


Рисунок 3 - Декомпозиция БП предоставлении услуг по физической подготовке

Входы – т.е. информация, подвергаемая обработке:

- «Оплата»
- «Клиент»

Выходы – т.е. данные, полученные в результате выполнения работы:

- «Отзыв»

Промежуточный объект - данные, которые служат входами для следующих функциональных блоков:

- «Членство»

Управление – условия, ограничения, указания:

- «Каталог членств»
- «Список тренеров»

- «Расписание»
- «Список тренировок и программ»

Механизмы или исполнители – те, кто выполняет работу:

- «Менеджер»
- «Тренер»

Глоссарий предметной области

Таблица 1 – Глоссарий предметной области

Название термина	Описание термина
Карта клиента	Карта клиента представляет собой документ, содержащий персональные данные клиента фитнес-центра, а также информацию о его членстве.
Список клиентов	Список клиентов представляет собой перечень всех зарегистрированных клиентов фитнес-центра.
Услуга	Предоставление разнообразных физических активностей, тренировок, оборудования и других ресурсов для улучшения физической формы, здоровья клиентов. Эти услуги могут включать в себя доступ к тренажерам, групповым занятиям, тренировкам с тренерами, массажу, саунам и другим активностям, направленным на поддержание и укрепление здоровья.
Сотрудник (менеджер)	Человек, который управляет операциями центра. Его обязанности включают управление персоналом, работу с клиентами. Менеджер играет ключевую роль в эффективной работе фитнес-центра и обеспечении удовлетворения клиентов.
Карта тренера	Карта тренера представляет собой документ, содержащую информацию о тренере.
Список тренеров	Список тренеров представляет собой компиляцию информации о специалистах, обучающий клиентов правильным методам выполнения упражнений и тренировочных программ, обеспечивая безопасность и эффективность занятий.

Членство	Форма партнерства или соглашения между клиентом и фитнес-центром, которое позволяет клиенту посещать центр и использовать его услуги, обычно за определенную плату и в определенные сроки. Членство может включать в себя доступ к тренировочному оборудованию, групповым занятиям, услугам тренера и другим фитнес-ресурсам, предоставляемым центром.
Каталог членств	Картотека членств, в которой представлено описание членства, с возможностью добавления новых, удаления, редактирования уже существующих.
Программа тренировок	Индивидуально разработанный план физических упражнений и активностей, который помогает клиенту достичь своих фитнес-целей. Эта программа учитывает уровень подготовки, цели и особенности клиента, определяет, какие упражнения, в каком порядке и с какой интенсивностью следует выполнять. Программа тренировок может также включать в себя советы по питанию и регулярное отслеживание прогресса.
Журнал заявок	Журнал заявок представляет собой систематизированную запись всех запросов и заявок от клиентов или посетителей фитнес-центра.
Список тренировок и программ	Список, предоставляющий информацию о всех тренировках и программах, которые доступны в фитнес-центре.

Видение

Совладельцы системы

Совладельцами данной системы являются:

- Менеджер фитнес-центра. Данный совладелец занимается административными задачами в фитнес-центре. Он осуществляет следующие функции: управление персоналом, учет клиентов, учет расписания, генерация отчетов о финансовой деятельности. Менеджер фитнес-центра выступает в качестве единственного пользователя системы и отвечает за эффективное управление всеми аспектами деятельности центра.
- Тренер. Данный совладелец работает с клиентами и проводит тренировки. Данный совладелец напрямую не взаимодействует с системой, а значит и не является её пользователем.

Границы системы

Поскольку данная система предназначена для использования одним пользователем, её границы определены довольно чётко исходя из области действия рабочего места менеджера фитнес-центра.

Описание проблемы предметной области:

Таблица 2 – Проблема в сложности выбора программы тренировок

Элемент	Описание
Проблема	Сложность в выборе программы тренировок
Воздействует на	Клиента
Результатом чего является	Недовольство клиента
Выигрыш от	Новой системы, направленной на решение данной проблемы
Может состоять в следующем	- облегчение для клиента поиска подходящей тренировки

	- улучшение процесса организации рейтинга тренировок - увеличение количества отзывов о программах тренировок
--	---

Управление операциями: многие фитнес-центры сталкиваются с ограниченной эффективностью в управлении своими операциями, что может привести к проблемам с планированием, контролем и мониторингом ресурсов.

Обслуживание клиентов: удовлетворенность клиентов и их лояльность часто страдают из-за неэффективной системы обслуживания, длительных ожиданий и недостаточной персональной поддержки.

Конкурентоспособность: в условиях растущей конкуренции фитнес-центрам необходимо продемонстрировать свою уникальность и привлекательность для клиентов.

Возможности системы:

1. Управление операциями:

- Автоматизация учета: Система будет обеспечивать автоматизацию процессов учета клиентов, сотрудников, расписания занятий и инвентаря.
- Управление ресурсами: Система будет предоставлять инструменты для оптимизации распределения ресурсов, включая сотрудников и оборудование.

2. Обслуживание клиентов:

- Персональная информация: Система будет хранить данные о клиентах, их фитнес-целях и предпочтениях для создания персонализированных программ тренировок и рекомендаций.

- Система бронирования: Клиенты смогут легко бронировать тренировки и услуги через веб-портал или мобильное приложение.

- Обратная связь и поддержка: Система предоставит механизмы для обратной связи с клиентами и предоставления им советов и поддержки.

3. Конкурентоспособность:

- Маркетинг и продвижение: Система будет интегрировать средства для эффективного маркетинга, акций и программ лояльности, чтобы привлекать и удерживать клиентов.

- Анализ данных: Система будет собирать и анализировать данные о клиентах и операциях для выявления тенденций и возможностей для улучшения бизнеса.

Этап Анализа

Концептуальная модель предметной области

Фрагмент концептуальной модели предметной области информационной системы фитнес-центра представлен на рисунке 4.



Рисунок 4 - Фрагмент концептуальной модели предметной области информационной системы фитнес-центра

Функциональные требования

Таблица 3 – Функциональные требования

Номер	Текст
FR010	Система должна позволять просматривать список сотрудников.
FR020	Система должна позволять просматривать список клиентов.
FR030	Система должна позволять вводить информацию о новом сотруднике.
FR040	Система должна позволять удалять информацию о сотруднике.
FR050	Система должна позволять обновлять информацию о сотруднике.
FR060	Система должна позволять вводить информацию о новом клиенте.
FR070	Система должна позволять удалять информацию о клиенте.
FR080	Система должна позволять обновлять информацию о клиенте.
FR090	Система должна позволять просматривать каталог услуг.
FR100	Система должна позволять вводить информацию о новых услугах.
FR110	Система должна позволять удалять информацию об услугах.
FR120	Система должна позволять обновлять информацию об услугах.
FR130	Система должна позволять просматривать каталог программ тренировок.
FR140	Система должна позволять вводить информацию о новых программах тренировок.
FR150	Система должна позволять удалять информацию о программах тренировок.
FR160	Система должна позволять обновлять информацию о

	программах тренировок.
FR170	Система должна позволять просматривать список тренеров.
FR180	Система должна позволять вводить информацию о новых тренерах.
FR190	Система должна позволять удалять информацию о новых тренерах.
FR200	Система должна позволять обновлять информацию о тренерах.
FR210	Система должна позволять при просмотре списка тренеров предусмотреть поиск по рейтингу.
FR220	Система должна позволять при просмотре списка услуг предусмотреть поиск по цене, рейтингу и другой информации.
FR230	Система должна иметь следующую структуру информации о сотруднике: <ul style="list-style-type: none"> ○ ФИО сотрудника ○ Дата рождения ○ Дата начала работы ○ Почта
FR240	Система должна иметь следующую структуру информации о клиенте: <ul style="list-style-type: none"> ○ ФИО клиента ○ Номер телефона ○ Почта ○ Дата рождения ○ Номер членства
FR250	Система должна иметь следующую структуру информации об услугах: <ul style="list-style-type: none"> ○ Цена ○ Описание

	<ul style="list-style-type: none"> ○ Количество продаж
FR260	<p>Система должна иметь следующую структуру информации о программах тренировок:</p> <ul style="list-style-type: none"> ○ Цена ○ Описание ○ Особенности
FR270	<p>Система должна иметь следующую структуру информации о тренере:</p> <ul style="list-style-type: none"> ○ ФИО тренера ○ Дата рождения ○ Фотография ○ Стаж ○ Тренировки, которые проводит
FR280	Система должна позволять проходить авторизацию.

Диаграммы вариантов использования

Примеры диаграмм вариантов использования информационной системы

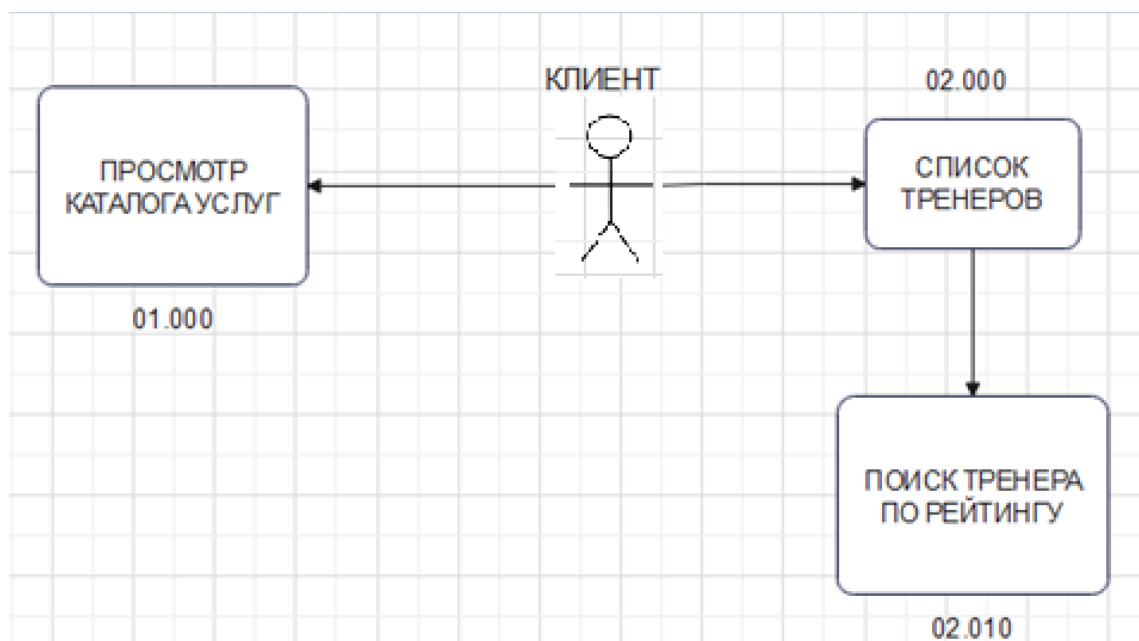


Рисунок 5 - Неполная диаграмма вариантов использования системы для клиента

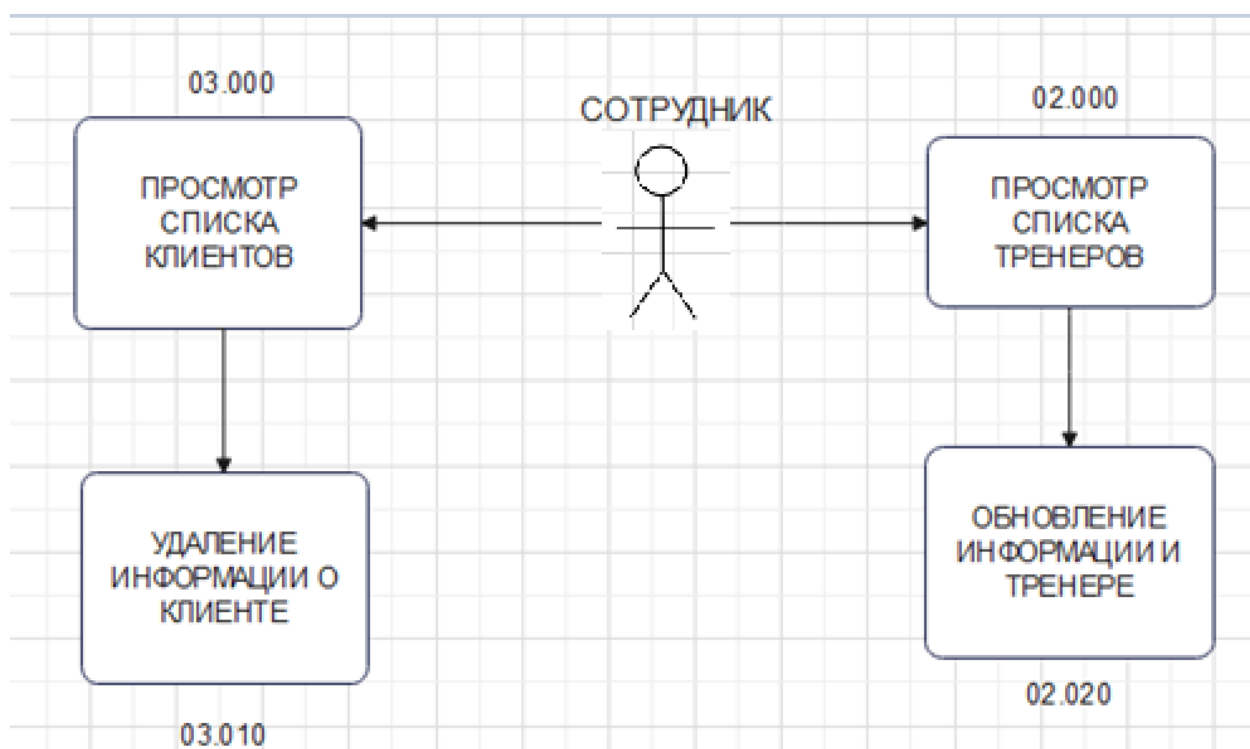


Рисунок 6 - Неполная диаграмма вариантов использования системы для сотрудника

Краткое и подробное описание вариантов использования

Просмотр списка сотрудников:

Описание: Только главный администратор имеет право открыть список со всеми сотрудниками, посмотреть информацию о них.

Требования: FR010, FR230

Основной ход событий: Главный администратор нажимает на соответствующую кнопку и перед ним открывается список сотрудников, работающих в фитнес-центре, в котором можно сделать поиск по ФИО или рейтингу.

Таблица 4 – Нормальный ход варианта использования просмотра списка сотрудников

Действия пользователя	Отклик системы
10. Администратор вызывает список сотрудников или вызывает функцию «Просмотреть каталог сотрудников» при просмотре результатов поиска.	20. Система отображает на экране отсортированным по ФИО список тренеров в виде: <ul style="list-style-type: none">• ФИО тренера• Вид тренировок• Квалификация• Стаж• Рейтинг
30. Сотрудник просматривает список тренеров	

Альтернативный ход событий: Происходит непредвиденный сбой при отображении списка сотрудников. Система отображает главному администратору сообщение об ошибке «Произошел непредвиденный сбой при отображении списка сотрудников. Список сотрудников не может быть

отображен». Главный администратор подтверждает прочтение сообщения. Список сотрудников отображается пустым. Выполнения варианта использования заканчивается.

Точки расширения:

Таблица 5 – Точки расширения просмотра списка сотрудников

Название	Описание	Вызываемый вариант использования
Форма поиска		02.010 Поиск определенного сотрудника
Элемент управления «Редактировать информацию»	При вызове сценария учитывается, какой сотрудник выбран в списке.	02.050 Изменение информации о сотруднике
Элемент управления «Новый сотрудник»		02.030 Ввод информации о новом сотруднике
Элемент управления «Удалить»	При вызове сценария учитывается, какой сотрудник выбран в списке.	02.040 Удаление информации о сотруднике

Просмотр списка клиентов:

Описание: Все сотрудники имеют право открыть список со всеми клиентами (сами клиенты не могут этого сделать ввиду конфиденциальности), посмотреть информацию о них.

Требования: FR020, FR240

Основной ход событий: Сотрудник нажимает на соответствующую кнопку и перед ним открывается список сотрудников, в котором можно сделать поиск по ФИО, номеру членства (либо же номеру заявки).

Таблица 6 – Нормальный ход варианта использования просмотра списка клиентов

Действия актера	Отклик системы
10. Сотрудник нажимает на соответствующую кнопку для вызова функции «Просмотреть список клиентов».	20. Система отображает список всех клиентов.
30. Сотрудник нажимает на соответствующую кнопку для сортировки по ФИО или номеру Сим-карты	40. Система отображает список всех клиентов, отсортированных по ФИО или номеру Сим-карты: <ul style="list-style-type: none"> ○ ФИО клиента ○ Пол ○ Дата рождения ○ Почта ○ Номер членства
50.Сотрудник просматривает список всех клиентов	

Альтернативный ход событий: Происходит непредвиденный сбой при отображении списка. Система отображает сообщение об ошибке «Произошел сбой. Список клиентов не может быть отображен». Сотрудник подтверждает прочтение сообщения. Список отображается пустым. Выполнения варианта использования заканчивается.

Точки расширения:

Таблица 7 – Точки расширения просмотра списка клиентов

Название	Описание	Вызываемый вариант использования
Форма поиска		04.020 Поиск определенного клиента
Элемент управления «Редактировать информацию»	При вызове сценария учитывается, какой клиент выбран в списке.	04.080 Изменение информации о клиенте
Элемент управления «Новый клиент»		04.060 Ввод информации о новом клиенте
Элемент управления «Удалить»	При вызове сценария учитывается, какой сотрудник выбран в списке.	04.070 Удаление информации о клиенте

Просмотр списка программ тренировок:

Описание: Все сотрудники имеют право открыть список со всеми программами тренировок, посмотреть цену, рейтинг.

Требования: FR130, FR260

Основной ход событий: Сотрудник нажимает на соответствующую кнопку и перед ним открывается список программ тренировок, в котором можно сделать поиск по цене или рейтингу.

Альтернативный ход событий: Происходит непредвиденный сбой при отображении списка. Система отображает сообщение об ошибке «Произошел сбой. Список не может быть отображен». Сотрудник подтверждает прочтение сообщения. Список отображается пустым. Выполнения варианта использования заканчивается.

Добавление новой услуги:

Описание: Данный прецедент может быть вызван при просмотре списка услуг. Этот прецедент описывает процесс добавления информации о новой услуге.

Требования: FR090, FR100, FR250

Основной ход событий: Главный администратор заполняет информацию о новой услуге, её цену, описание.

Альтернативный ход событий: Главный администратор удаляет созданную услугу. Выполнение сценария на этом заканчивается.

Данные введены некорректно. Система выдает сообщение об ошибке. Главный администратор подтверждает прочтение сообщения. Выполнение сценария на этом заканчивается.

Главный администратор не имеет право создавать новую услугу, так как она уже существует. Система выдает сообщение об ошибке. Главный администратор подтверждает прочтение сообщения. Выполнение сценария на этом заканчивается.

Проектирование пользовательского интерфейса

Дерево функций представляет собой иерархическую структуру действий, реализованных в информационной системе. Состав и классификация функций разрабатываемой системы представлены в виде дерева функций на рисунке 7:

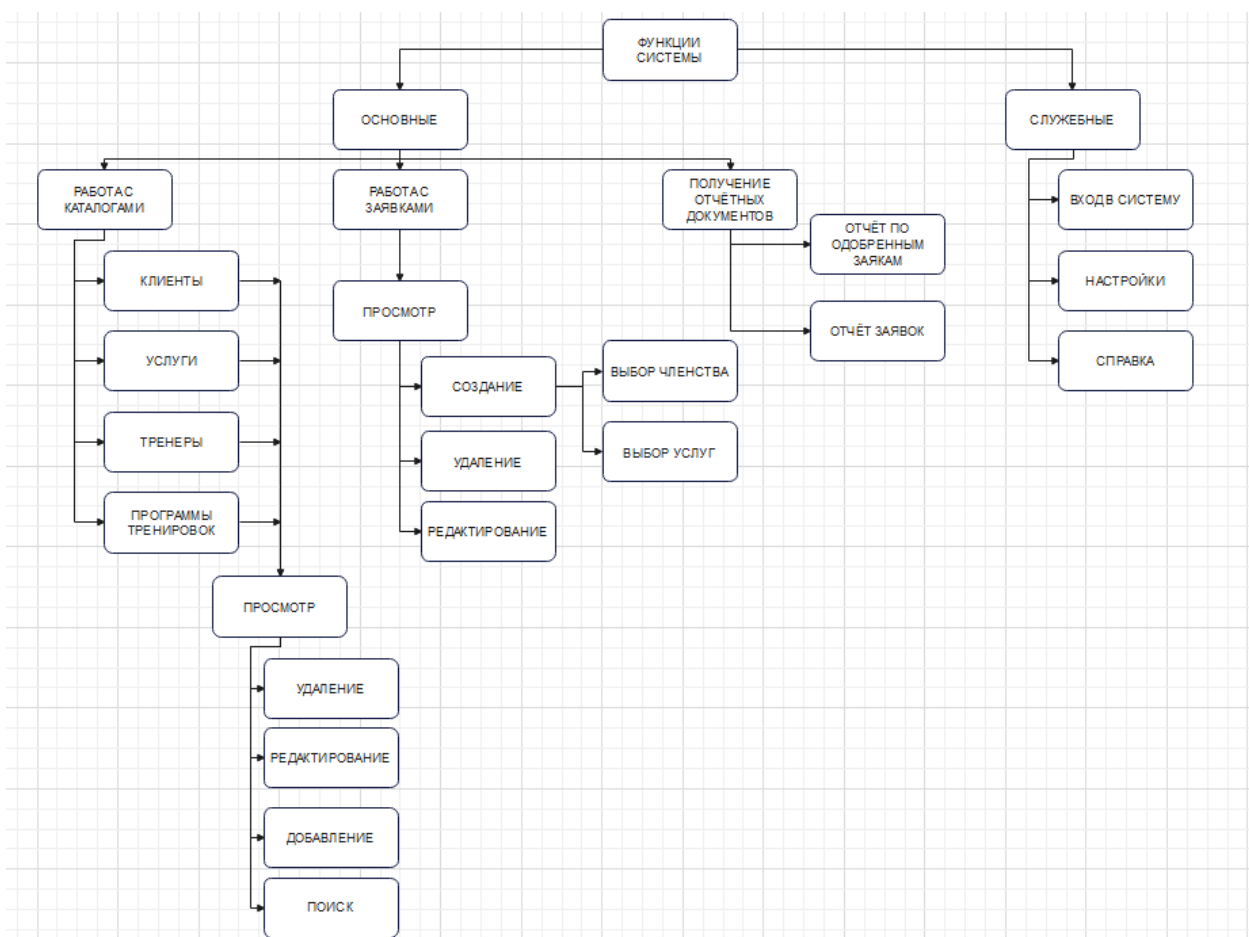


Рисунок 7 - Пример дерева функций информационной системы

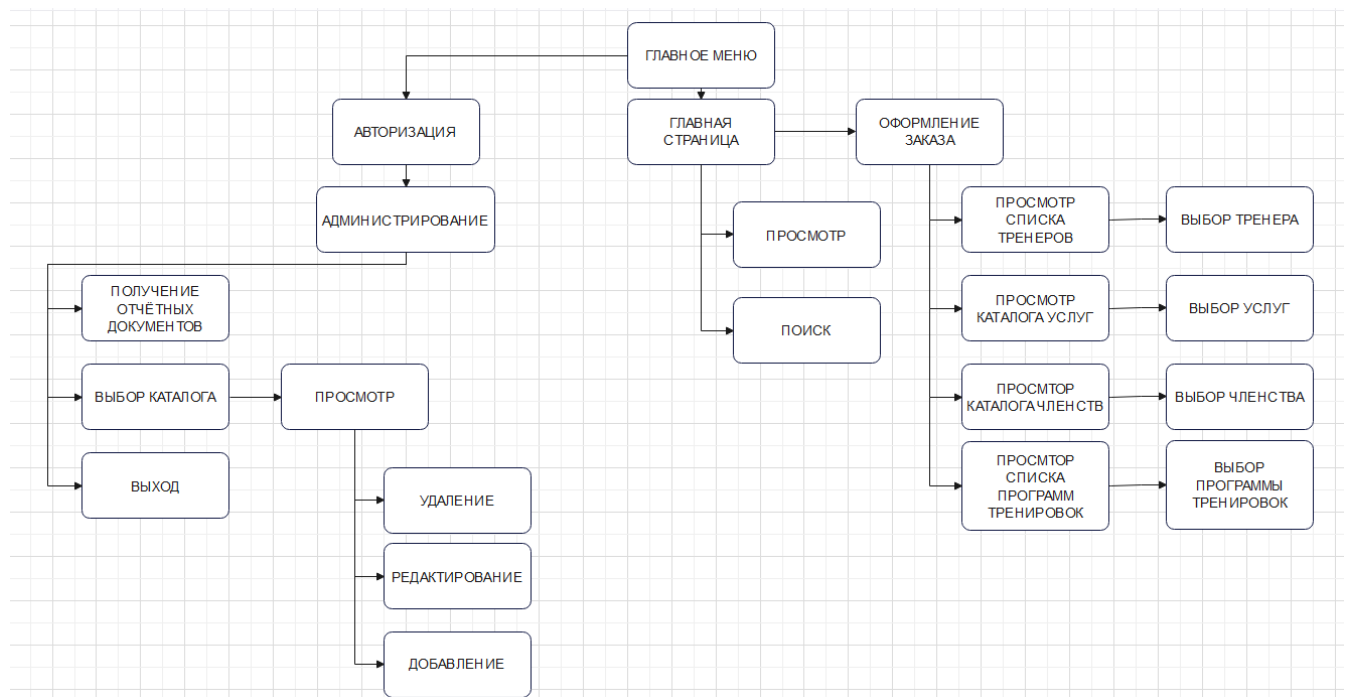


Рисунок 8 - Пример сценария диалога с информационной системой

Примеры макетов интерфейсов

Технически система может состоять, например, из разделов:

- Пользовательский (режим просмотра);
- Административный (режим редактирования);

Фитнес-центр Power			
Главное меню	Элемент меню	Элемент меню	Элемент меню
<div><div><div>Авторизация</div><div>Логин: <input type="text"/></div><div>Пароль: <input type="password"/></div><div>Вход</div></div><div>Содержимое выбранного раздела</div></div>			

Рисунок 9 - Пример макета страницы пользовательского раздела

Административный раздел					
Выберите опцию:					
Редактирование 1	Редактирование 2	Редактирование 3	Редактирование 4	...	Редактирование n

Рисунок 10 - Пример макета страницы административного раздела

При выборе одной из опций появляется окно редактирования, например, изменения или добавления данных в базу данных (см. рисунок 11)

Административный раздел						
Редактирование таблицы N						
	Поле 1	Поле 2	Поле 3	Поле 4	Поле 5	

Форма для добавления записи	
Поле 1	
Поле 2	
Поле 3	
Поле 4	
Поле 5	
<div>Назад</div>	

Рисунок 11 - Пример макета страницы административного раздела подраздела редактирования

Этап проектирования

Диаграммы анализа

На диаграмме анализа изображают:

- объекты, участвующие в варианте использования,
- способы взаимодействия этих объектов.

Это диаграмма, на которой вместо обычных в UML символов классов изображаются пиктограммы трех видов, представленных на рисунке ниже. Граничные объекты актеры используют для взаимодействия с системой. Сущностные объекты – это обычно объекты из модели предметной области. Управляющие объекты (обычно называются контроллерами, так как им ничего не соответствует в реальном мире) выполняют функции «клея» между граничными и сущностными объектами.

На рисунке 12 приведен иллюстративный пример диаграммы анализа для варианта использования «Изменение информации о тренере 02.000»

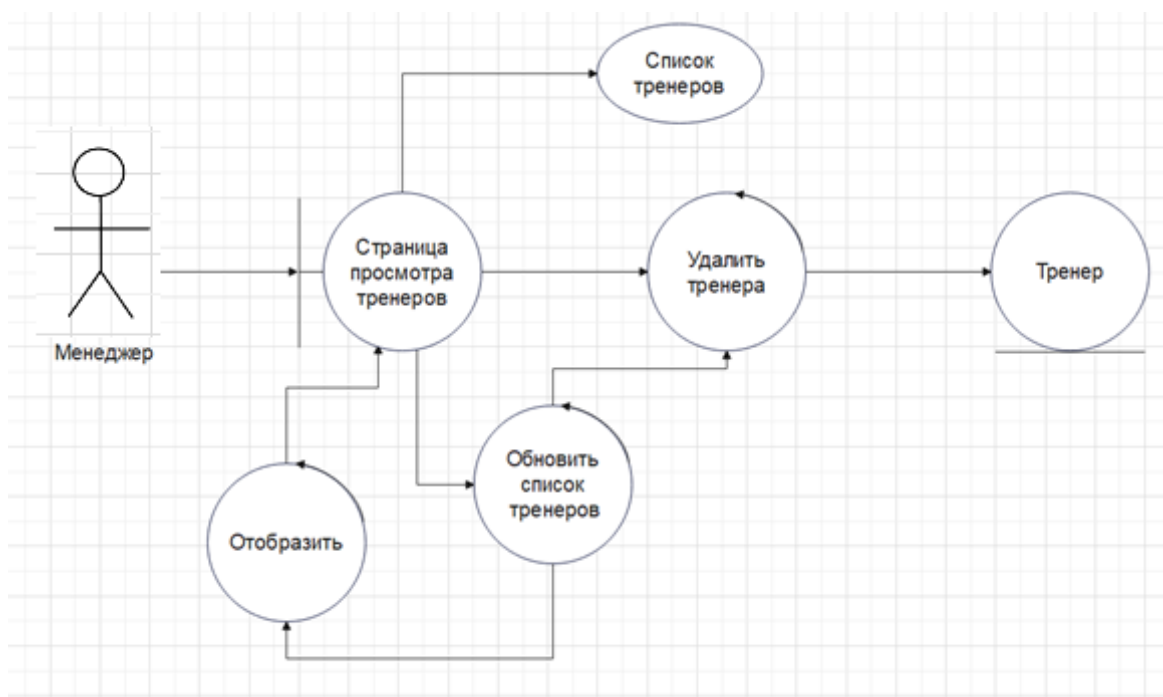


Рисунок 12 - Диаграммы анализа варианта использования «Изменение информации о тренере 02.000»

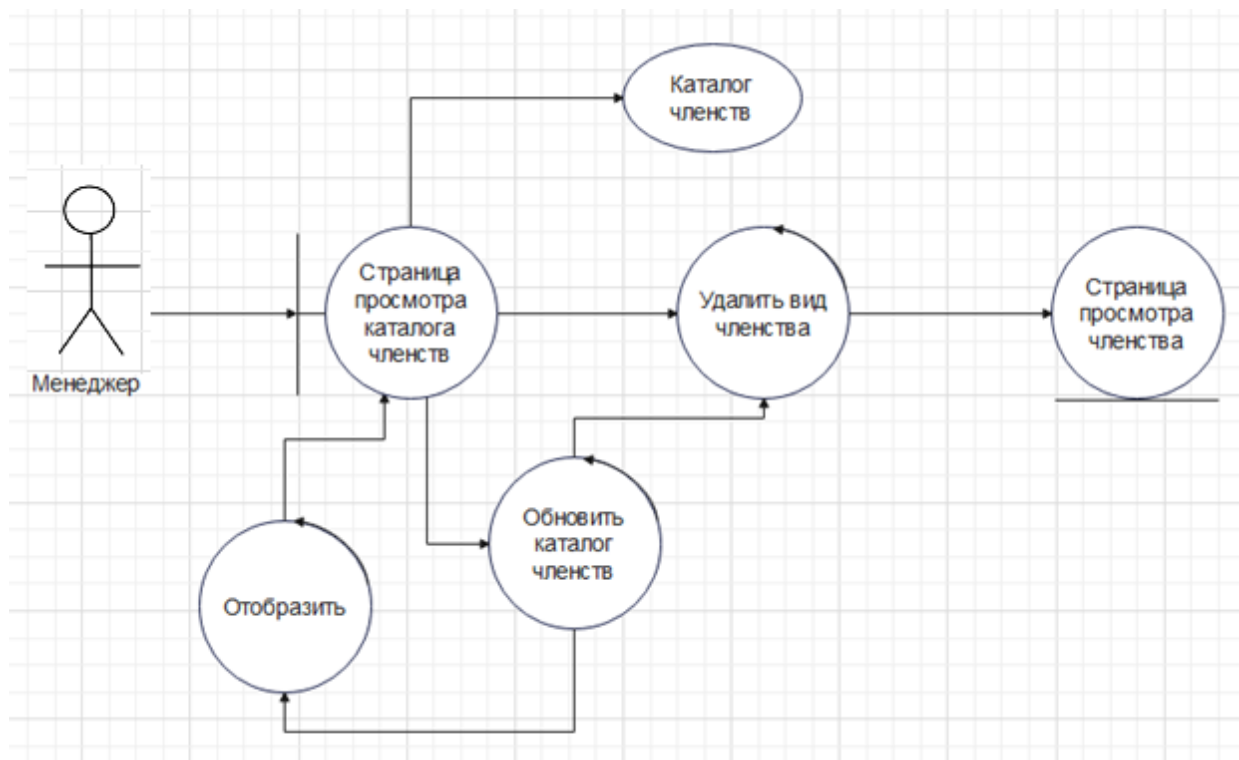


Рисунок 13 - Диаграммы анализа варианта использования «Изменение информации о членстве 02.000»

Диаграмма последовательности

Диаграммы последовательности отражают поток событий (бизнес-логику), происходящих в рамках варианта использования. С помощью диаграмм взаимодействия можно определить классы, которые нужно создать, связи между ними, а также операции и ответственности каждого класса. Диаграммы последовательностей состоят из последовательных откликов системы на различные действия пользователя, взаимодействия отдельных элементов системы между собой и вариантами ответа за запросы. Диаграммы последовательности упорядочены по времени. Они полезны для того, кто хочет понять логическую последовательность событий в сценарии.

Пример диаграммы последовательности для варианта использования «Просмотр списка программ тренировок 04.000» представлен на рисунке 14.

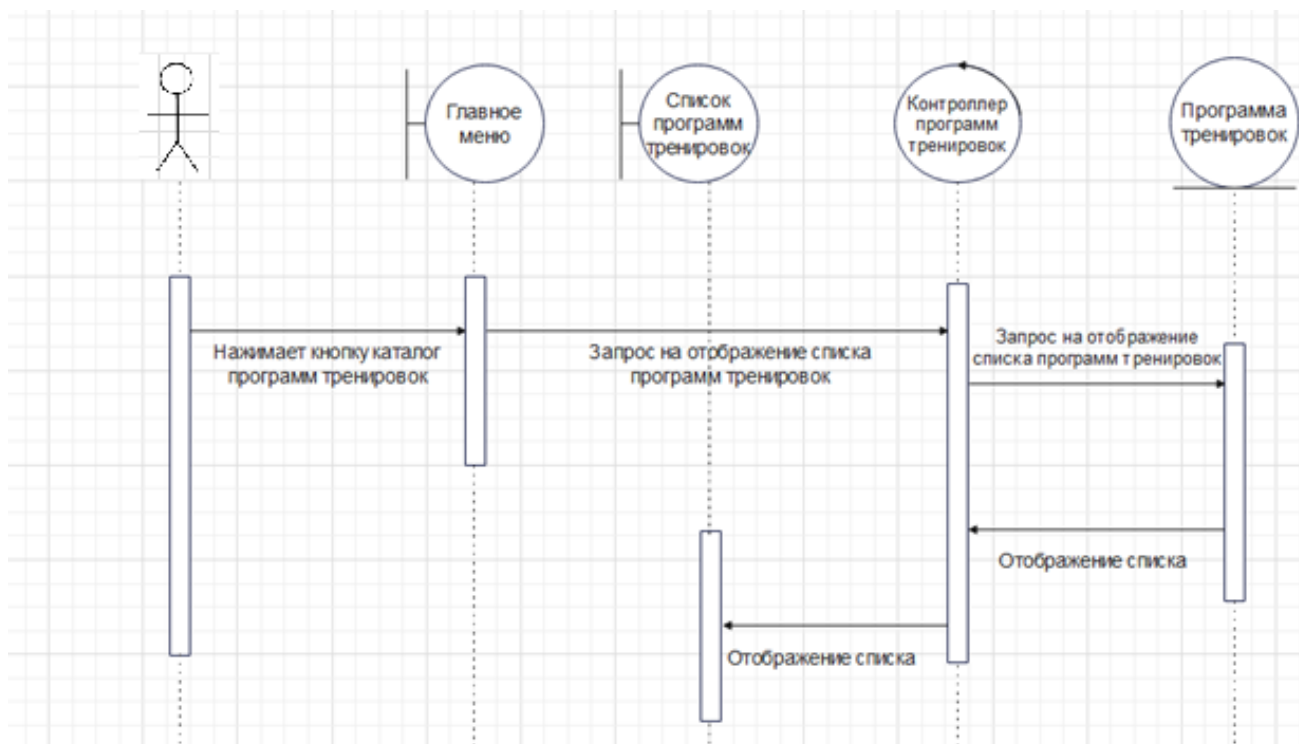


Рисунок 14 - Диаграмма последовательности для прецедента «Просмотр списка программ тренировок 04.000»

Для остальных объектов диаграммы последовательности аналогичны.

Диаграммы классов этапа проектирования

Диаграмма классов этапа проектирования дополняет и расширяет концептуальную модель. На этой диаграмме для каждого отображаемого на диаграмме класса идентифицируются и специфицируются атрибуты (данные), а из диаграмм последовательности выявляются и специфицируются операции (методы) и их параметры. Таким образом, диаграмма классов этапа проектирования содержит в себе детальную информацию по всем входящим в диаграмму классам. Поэтому для лучшей визуализации эту диаграмму рекомендуется представлять в пакетном виде. Напомним, что, имея в виду язык UML, пакеты - это логические "ящики", по которым могут быть разложены, в частности, объединенные в группы классы диаграммы классов этапа проектирования. Поскольку создаваемая Вами программная система должна иметь трехуровневую архитектуру, то рекомендуем Вам сначала выделить пакеты по типу их принадлежности к одному из следующих трех уровней: уровню представления (граничные классы), уровню приложения (контроллеры), уровню хранения данных (сущностные классы).

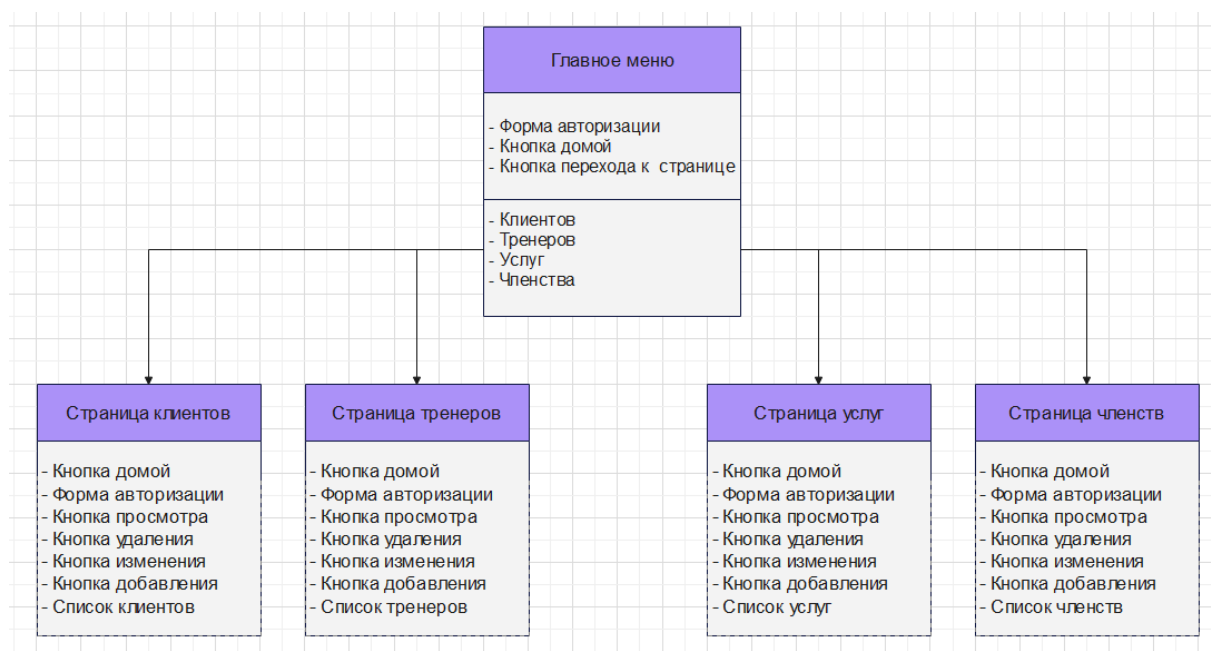


Рисунок 15 - Диаграмма классов уровня представления

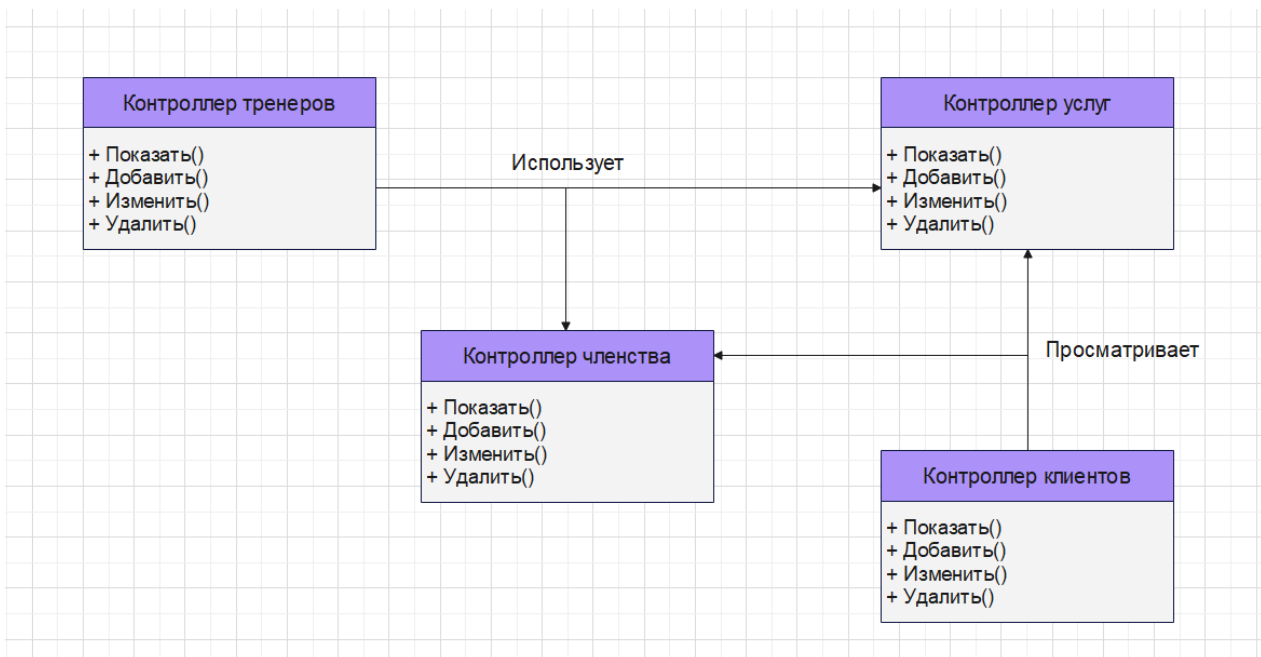


Рисунок 16 - Диаграмма классов прикладного уровня

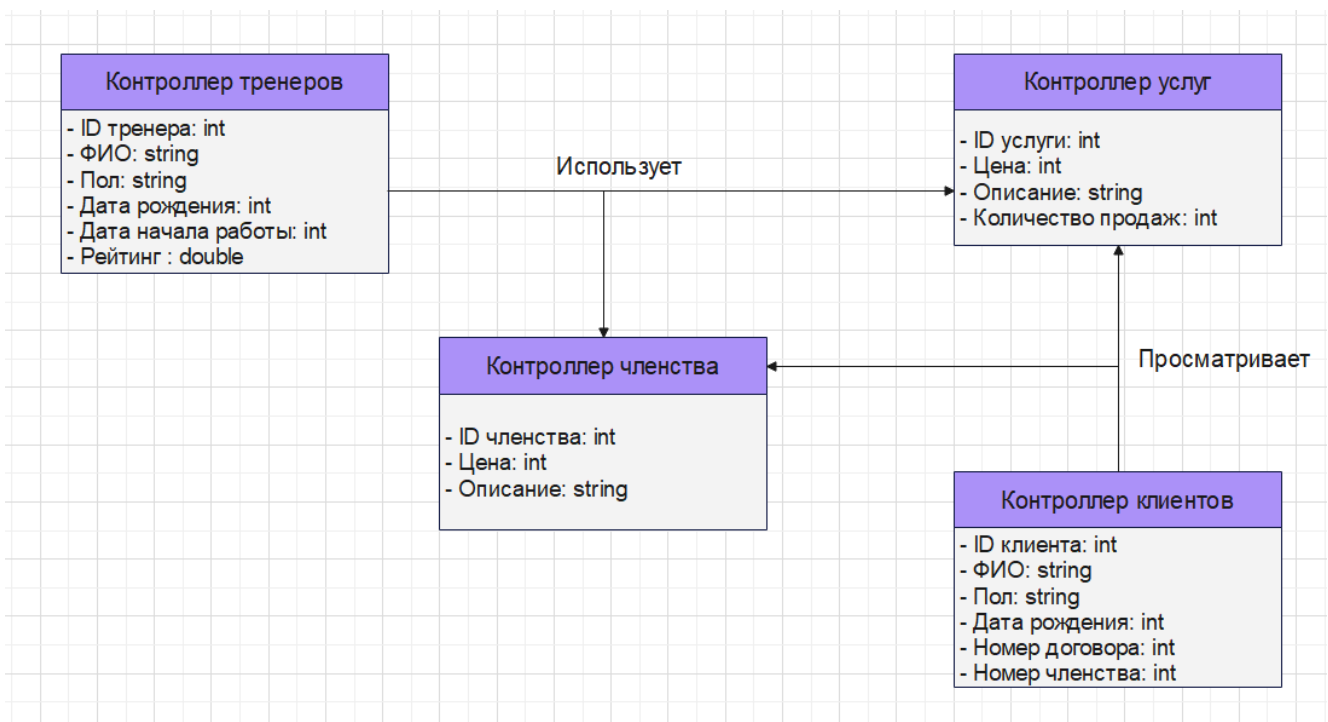


Рисунок 17 - Диаграмма классов уровня хранения данных (базы данных)

Схема (структура) базы данных

Нужно произвести нормализацию диаграммы классов, по которой будет создаваться модель базы данных. Т.е. надо проанализировать все ассоциации между классами и заменить связи "много-ко-многим" и "один-к одному", разбив или объединив таблицы соответственно.

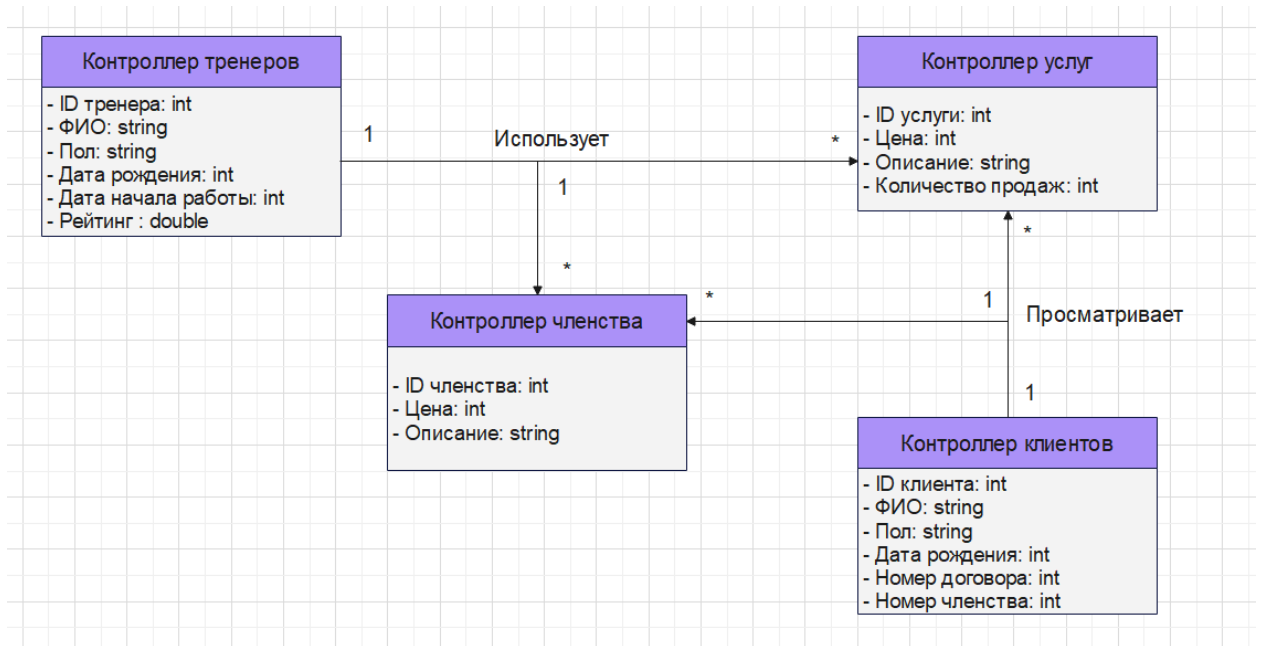


Рисунок 18 - Физическая модель базы данных.

Конструкторская часть

Введение

Данный проект направлен на разработку современной информационной системы для фитнес-центра "EpicPower". Основной целью проекта является создание инновационной платформы, предназначенной для эффективного управления операционными процессами и улучшения обслуживания клиентов. Задачей проекта является разработка интуитивно понятного и удобного пользовательского интерфейса как для сотрудников фитнес-центра, так и для конечных пользователей.

Система "EpicPower" стремится обеспечить простоту и оперативность в управлении всеми аспектами жизни фитнес-центра, сокращая временные затраты и улучшая общее взаимодействие с клиентами. Целью данного проекта является создание цифрового решения, которое оптимизирует процессы записи на тренировки, отслеживания занятий и повышения мотивации клиентов для достижения своих фитнес-целей.

Основания для разработки

Учебный план направления “Информационный системы и телекоммуникации”, утвержденный Ректором МГТУ Гординым М.В.

Требования к функциональным характеристикам

Таблица 8 – Функциональные требования

Номер	Текст
FR010	Система должна позволять просматривать список сотрудников.
FR020	Система должна позволять просматривать список клиентов.
FR030	Система должна позволять вводить информацию о новом сотруднике.
FR040	Система должна позволять удалять информацию о сотруднике.

FR050	Система должна позволять обновлять информацию о сотруднике.
FR060	Система должна позволять вводить информацию о новом клиенте.
FR070	Система должна позволять удалять информацию о клиенте.
FR080	Система должна позволять обновлять информацию о клиенте.
FR090	Система должна позволять просматривать каталог услуг.
FR100	Система должна позволять вводить информацию о новых услугах.
FR110	Система должна позволять удалять информацию об услугах.
FR120	Система должна позволять обновлять информацию об услугах.
FR130	Система должна позволять просматривать каталог программ тренировок.
FR140	Система должна позволять вводить информацию о новых программах тренировок.
FR150	Система должна позволять удалять информацию о программах тренировок.
FR160	Система должна позволять обновлять информацию о программах тренировок.
FR170	Система должна позволять просматривать список тренеров.
FR180	Система должна позволять вводить информацию о новых тренерах.
FR190	Система должна позволять удалять информацию о новых тренерах.
FR200	Система должна позволять обновлять информацию о тренерах.
FR210	Система должна позволять при просмотре списка тренеров

	предусмотреть поиск по рейтингу.
FR220	Система должна позволять при просмотре списка услуг предусмотреть поиск по цене, рейтингу и другой информации.
FR230	Система должна иметь следующую структуру информации о сотруднике: <ul style="list-style-type: none"> ○ ФИО сотрудника ○ Дата рождения ○ Дата начала работы ○ Почта
FR240	Система должна иметь следующую структуру информации о клиенте: <ul style="list-style-type: none"> ○ ФИО клиента ○ Номер телефона ○ Почта ○ Дата рождения ○ Номер членства
FR250	Система должна иметь следующую структуру информации об услугах: <ul style="list-style-type: none"> ○ Цена ○ Описание ○ Количество продаж
FR260	Система должна иметь следующую структуру информации о программах тренировок: <ul style="list-style-type: none"> ○ Цена ○ Описание ○ Особенности
FR270	Система должна иметь следующую структуру информации о тренере: <ul style="list-style-type: none"> ○ ФИО тренера ○ Дата рождения

	<ul style="list-style-type: none"> ○ Фотография ○ Стаж ○ Тренировки, которые проводит
FR280	Система должна позволять проходить авторизацию.

Требования к надежности

1. Повышение надежности ввода данных:

- **Четкие условия ввода:** Внедрение четких и подробных правил для вводимых значений, определение диапазонов и форматов данных.
- **Применение масок:** Введение масок для контроля за правильностью ввода, обеспечивая точность внесения информации и предотвращая возможные ошибки.
- **Механизмы двойного ввода:** Реализация системы двойного ввода для предотвращения случайных ошибок и обеспечения более высокой точности данных.
- **Использование всплывающих списков:** Внедрение списков выбора для предоставления пользователям ограниченного, но правильного выбора значений, что способствует более точному вводу.

2. Защита информации от разрушения:

- **Защита от потери данных при выключении:** Разработка механизмов, которые обеспечивают сохранение состояния данных перед выключением питания, минимизируя риски потери информации.
- **Автозапись данных:** Внедрение автоматизированной системы записи данных для предотвращения возможной утраты информации при сбоях в работе системы.
- **Механизмы отката (rollback):** Реализация возможности отката к предыдущему состоянию системы в случае обнаружения ошибок, обеспечивая стабильность функционирования.

- **Журнальные файлы:** Введение поддержки журнальных файлов для регистрации действий и событий в системе, что облегчит анализ и исправление возможных проблем.

- **Архивирование файлов:** Применение процесса архивирования файлов для создания резервных копий данных и обеспечения возможности быстрого восстановления в случае необходимости.

Условия эксплуатации

Условия эксплуатации для информационной системы «EpicPower» на базе Django с использованием SQLite:

1. Технические требования:

- Система предназначена для работы на сервере с установленной операционной системой, поддерживаемой Django.

- Требуется установленный Python в соответствии с требованиями фреймворка.

- Доступ к базе данных SQLite для записи и чтения данных.

- Рекомендуется использование современных браузеров (Google Chrome, Mozilla Firefox, Microsoft Edge).

2. Режим работы:

- Проект разработан с учетом 24/7 режима работы.

- Плановое техническое обслуживание проводится в периоды минимальной активности.

3. Технические характеристики помещения:

- Минимальная площадь офиса для размещения сервера.

- Допустимый температурный режим: 18–25°C.

- Освещение соответствует стандартам офисных помещений.
- Влажность поддерживается в пределах 40–60%.

4. **Обслуживание и квалификация персонала:**

- Система разработана для обслуживания без специального образования.
- Рекомендуется наличие персонала с базовыми навыками работы с Django и управления базой данных SQLite.
- Пользовательский интерфейс ориентирован на простоту использования, с дружественными материалами и подсказками.

Требования к составу и параметрам технических средств

1. **Серверы и Хранение данных:**

- Совместимость с Django и SQLite.
- Минимальные характеристики: Intel Xeon E3, 8 ГБ ОЗУ, 256 ГБ SSD.
- Регулярные бэкапы базы данных.
- Интеграция с облачными решениями для дополнительной устойчивости данных.

2. **Рабочие места:**

- Совместимость с операционной системой (например, Windows 10).
- Минимальные характеристики: Intel Core i5, 8 ГБ ОЗУ, 256 ГБ SSD.
- Поддержка веб-интерфейса.

3. **Сетевая Инфраструктура:**

- Стабильное и безопасное сетевое взаимодействие.
- Минимальная пропускная способность сети должна соответствовать объему передаваемых данных и обеспечивать минимальные задержки.
- Развертывание системы мониторинга для оперативного выявления и решения сетевых проблем.

4. **Телекоммуникационное Оборудование:**

- Совместимость с инфраструктурой фитнес-центра.

Требования к информационной и программной совместимости

1. Программная Среда:

- **Операционные системы:** Поддержка Windows, macOS, Linux, а также мобильных платформ Android и iOS.
- **Браузеры:** Совместимость с Google Chrome, Mozilla Firefox, Safari, Microsoft Edge.

2. Протоколы Интерфейсов:

- **Сетевые протоколы:** Поддержка TCP/IP, HTTP, HTTPS для безопасной и эффективной связи с серверами и клиентами.

3. Требования к Структуре Информации:

- **Входная Информация:** Поддержка форматов данных для регистрации клиентов, обработки заявок на услуги, и взаимодействия с клиентской информацией, например, JSON, XML.
- **Выходная Информация:** Генерация отчетов, счетов и других документов в структурированных форматах для предоставления информации клиентам и внутреннего пользования.

Требования к маркировке и упаковке

Нет

Требования к транспортировке и хранению

Нет

Требования к программной документации

1. Инструкция для Пользователя:

- **Подготовка подробной инструкции:**

- Охват всех основных функций и возможностей фитнес-системы, включая регистрацию, выбор тренировок.
- Детальное описание шагов по регистрации новых клиентов, обработке запросов на изменение услуг и других рутинных операций.
- Предоставление информации по устранению распространенных проблем, таких как восстановление паролей, и вариантов получения технической поддержки.

2. **Техническая Документация:**

- **Создание технической документации для администраторов:**
- Подробная информация о конфигурации серверов, включая требования к аппаратному обеспечению и настройкам сетевых элементов.
- Документирование процессов обновления и обслуживания системы, включая рекомендации по резервному копированию и восстановлению данных.
- Обеспечение документации по масштабированию системы для удовлетворения растущих потребностей фитнес-центра.

Технико–экономические показатели

Технико-экономические показатели для фитнес-центра "EpicPower" представляют собой комплексный анализ. Внедрение инновационной информационной системы в фитнес-центр означает стремление к оптимизации операционных процессов. Предполагается, что использование современных технологий позволит существенно снизить расходы на бумажную документацию, сделав внутренние процессы более эффективными и экономичными. Это, в свою очередь, создаст основу для повышения качества обслуживания клиентов, обеспечивая им более удобный и эффективный опыт взаимодействия с фитнес-центром.

Прогнозируется, что внедрение новой системы будет сопровождаться увеличением прибыли. Этот рост прибыли ожидается за счет привлечения новых клиентов, а также предоставления им дополнительных услуг и персонализированных подходов. Система сможет адаптироваться к потребностям клиентов, создавая дополнительные источники дохода через разнообразные программы, дополнительные тренировки или продукты.

Таким образом, технико-экономические показатели подчеркивают стратегическую значимость внедрения современных информационных решений в фитнес-центре " EpicPower" для достижения не только экономической эффективности, но и укрепления позиций на рынке, обогащения предоставляемых услуг и улучшения взаимодействия с клиентами.

Стадии и этапы разработки

Таблица 9 – Стадии и этапы разработки

Этап №	Содержание	Срок окончания	Отчёт
1	Подготовка	10.10.23	Описание предметной области, гlossарий, функциональная модель бизнес- процессов
2	Анализ	28.10.23	Концептуальная модель предметной области, технические требования, диаграммы вариантов использования

3	Проектирование	14.11.23	Диаграммы анализа, последовательности, этапов проектирования, база данных
4	Создание	20.12.23	Рабочий сайт фитнес-центра

Архитектура системы

Усилия в разработке архитектуры системы "EpicPower" направлены на обеспечение не только эффективной, но и удобной работы для пользователей. Система стремится соответствовать ряду важных требований, включая масштабируемость, гибкость, надежность и безопасность.

Масштабируемость и Гибкость: Система готова к расширению, обеспечивая возможность быстрого адаптирования к изменениям в потребностях фитнес-центра. Гибкая архитектура позволяет оперативно внедрять изменения, минимизируя трудозатраты.

Надежность и Безопасность: Особое внимание уделяется надежности и безопасности системы. Принимаются меры по минимизации времени простоя и обеспечению защиты данных от несанкционированного доступа и кибератак.

Трехуровневая архитектура:

Реализация приложения осуществляется с использованием трехуровневой архитектуры, включающей в себя уровень представления (View), уровень приложения (Controller) и уровень хранения данных (Model).

1. Модель (Model):

- *Назначение:* Модель определяет структуру приложения и управляет данными. В контексте фитнес-центра "EpicPower" это включает в себя информацию о клиентах, записях тренировок, расписании занятий и других аспектах.

- *Функции:*
- Определение моделей данных в файле `models.py`.
- Взаимодействие с базой данных SQLite, используя ORM фреймворк Django.
- Установление связей между объектами для эффективного хранения и обработки данных.

2. Представление (View):

- *Назначение:* Представление создает пользовательский интерфейс и обеспечивает взаимодействие с ним. Это то, что видит и использует конечный пользователь.

- *Функции:*
- Использование Django Templates для формирования HTML-страниц, динамически заполняемых данными из базы данных.
- Поддержка CSS и JavaScript для улучшения пользовательского интерфейса.
- Обработка пользовательских запросов и взаимодействие с контроллером.

3. Контроллер (Controller):

- *Назначение:* Контроллер связывает модель и представление, обеспечивает логику приложения и проверяет корректность взаимодействия между моделью и представлением.

- *Функции:*

- Определение функций в файле `views.py`, обрабатывающих HTTP-запросы и взаимодействующих с моделью.
- Регистрация, авторизация и аутентификация пользователей.
- Валидация вводимых данных и обработка форм.

Дополнительные аспекты:

- *Административная Панель:* Уровень представления включает административную панель для управления объектами, что упрощает администрирование данных.
- *Настройки (Settings):* Конфигурация базы данных и других аспектов приложения определена в файле `settings.py`, обеспечивая прозрачное взаимодействие с базой данных.

Такая трехуровневая архитектура позволяет эффективно управлять сложностью системы, делает ее более гибкой и поддерживаемой, а также обеспечивает четкое разделение функциональности между ее компонентами.

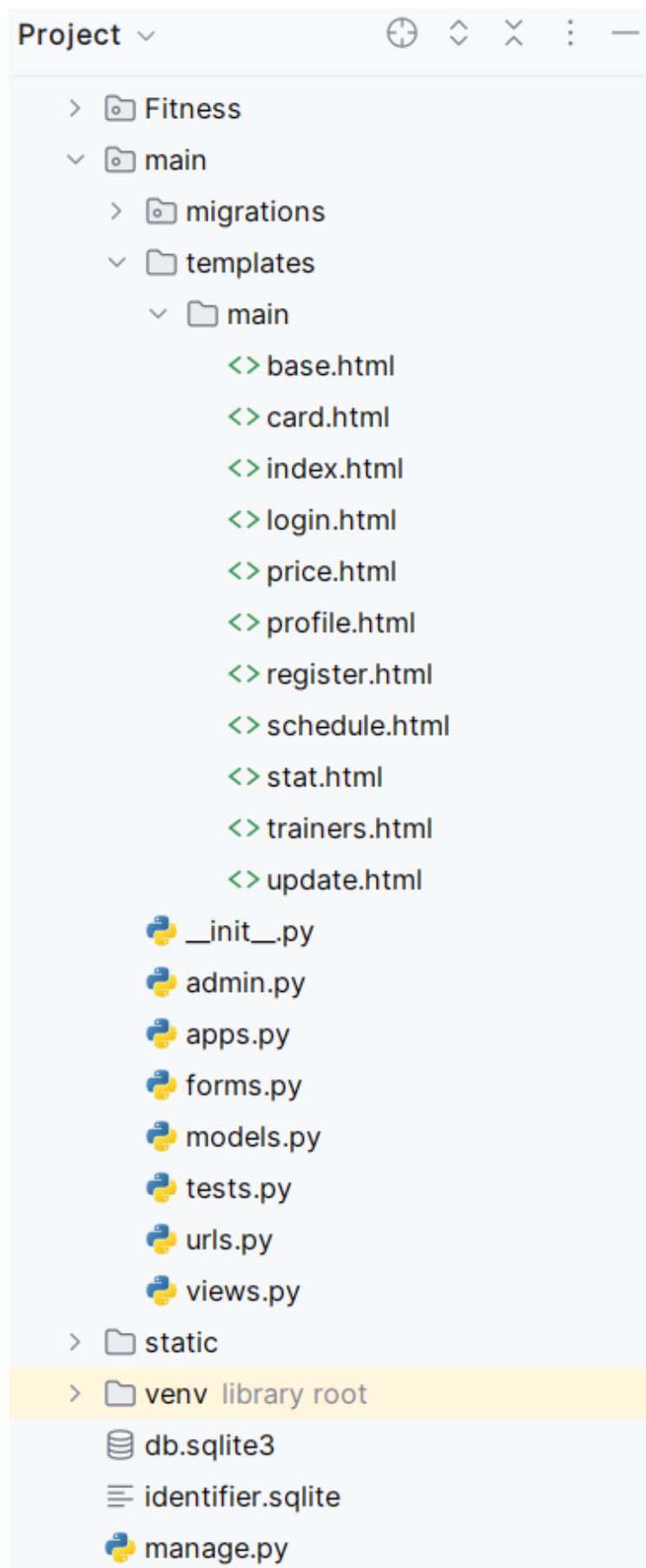


Рисунок 19 – Структура проекта

База данных

В проекте используется СУБД SQLite, исходя из следующих преимуществ:

- *Легкость внедрения:* Без необходимости сервера, SQLite обеспечивает удобное хранение данных в одном файле, упрощая процесс развертывания.
- *Кросс-платформенность:* Поддержка на различных ОС обеспечивает гибкость в разработке и установке приложения.
- *Экономия ресурсов:* SQLite эффективен для небольших проектов, минимизируя системные требования, что важно для среднего по размеру проекта.
- *Простота использования:* Отсутствие сложной установки и конфигурации сервера делает SQLite удобным для нашего проекта.
- *Хранение в одном файле:* Данные компактно упакованы в одном файле, что обеспечивает удобное управление информацией и её резервное копирование.

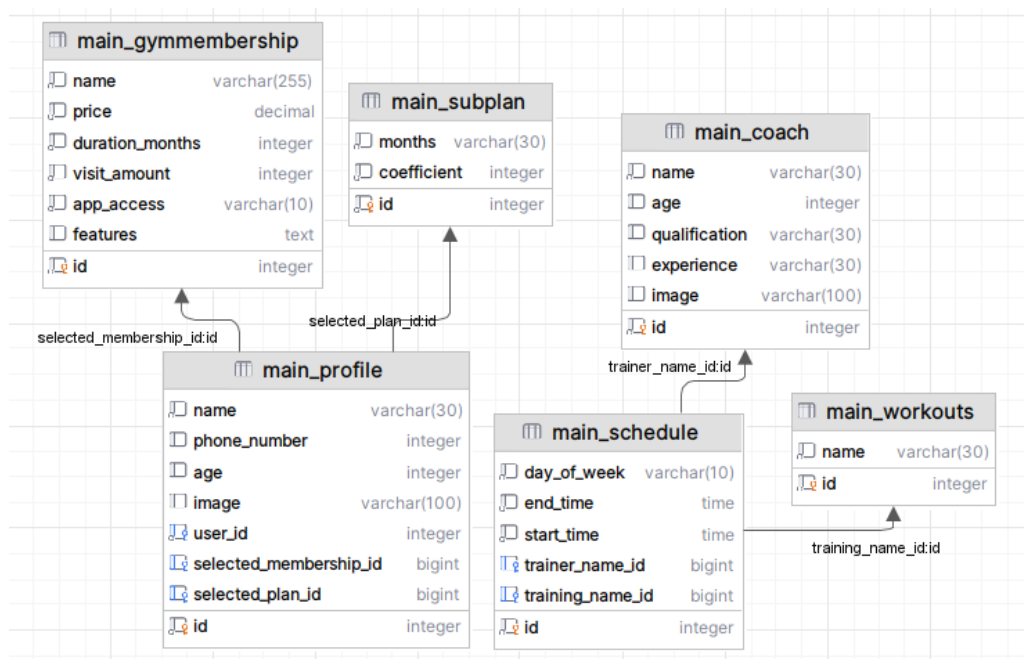


Рисунок 20 – Схема базы данных

```

class GymMembership(models.Model):
    name = models.CharField(max_length=255, verbose_name='Название абонемент')
    price = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Цена')
    duration_months = models.IntegerField(verbose_name='Продолжительность (месяцев)')
    visit_amount = models.IntegerField(verbose_name='Количество посещений в неделю', default=0)
    app_access = models.CharField(*args: 'Доступ к приложению с онлайн-тренировками', max_length=10, default="Нет")
    features = models.TextField(blank=True, null=True, verbose_name='Особенности абонемента')

    def __str__(self):
        return self.name

class Meta:
    verbose_name = 'Членство'
    verbose_name_plural = 'Членства в фитнес-центре'
  
```

Рисунок 21 – Описание таблицы GymMembership

```

class SubPlan(models.Model):
    months = models.CharField(*args: 'Количество месяцев', max_length=30)
    coefficient = models.IntegerField(verbose_name='Коэффициент')

    def str(self):
        return f'{self.months} месяцев, Коэффициент: {self.coefficient}'

class Meta:
    verbose_name = 'План членства'
    verbose_name_plural = 'Планы членств'
  
```

Рисунок 22 – Описание таблицы SubPlan

```

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=30, blank=True)
    phone_number = models.IntegerField(null=True, blank=True)
    age = models.IntegerField(null=True, blank=True)
    image = models.ImageField(null=True, blank=True, upload_to='coachs/')
    selected_membership = models.ForeignKey(GymMembership, on_delete=models.SET_NULL, null=True, blank=True)
    selected_plan = models.ForeignKey(SubPlan, on_delete=models.SET_NULL, null=True, blank=True)

    def __str__(self):
        return self.name

class Meta:
    verbose_name = 'Профиль'
    verbose_name_plural = 'Профили'

```

Рисунок 23 – Описание таблицы Profile

```

class Schedule(models.Model):
    day_of_week = models.CharField(max_length=10, default='День')
    start_time = models.TimeField(default='23:59:59')
    end_time = models.TimeField(default='23:59:59')
    trainer_name = models.ForeignKey(Coach, on_delete=models.SET_NULL, blank=True, null=True)
    training_name = models.ForeignKey(Workouts, on_delete=models.SET_NULL, blank=True, null=True)

    def __str__(self):
        return f"{self.trainer_name} - {self.day_of_week} - {self.start_time}-{self.end_time} - {self.training_name}"

class Meta:
    verbose_name = 'Расписание'
    verbose_name_plural = 'Расписания'

```

Рисунок 24 – Описание таблицы Schedule

```

class Coach(models.Model):
    name = models.CharField(*args: 'Имя тренера', max_length=30)
    age = models.IntegerField(verbose_name: 'Возраст', null=True, blank=True)
    qualification = models.CharField(*args: 'Специализация', max_length=30, null=True, blank=True)
    experience = models.CharField(*args: 'Опыт работы', max_length=30, null=True, blank=True)
    image = models.ImageField(null=True, blank=True, upload_to='static/assets/css/img')

    def __str__(self):
        return self.name

class Meta:
    verbose_name = 'Тренер'
    verbose_name_plural = 'Тренеры'

```

Рисунок 25 – Описание таблицы Coach

```
class Workouts(models.Model):
    name = models.CharField(*args: 'Название тренировки', max_length=30)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = 'Тренировка'
        verbose_name_plural = 'Тренировки'
```

Рисунок 26 – Описание таблицы Workouts

Серверная часть системы

Для разработки серверной части были выбраны Django и Python. Django обеспечивает быструю разработку и поддерживает структуру MVC, а Python предоставляет читаемость и гибкость.

Описание бизнес-логики системы:

1. Управление тренировочными программами:

- Модуль создания, редактирования и удаления программ с отслеживанием их статуса и параметров.

2. Управление членствами:

- Модуль создания и редактирования членств с отслеживанием их статуса и сроков действия.

3. Обработка запросов от клиентов:

- Модуль обработки запросов на запись на тренировки и изменение параметров членств.

4. Управление клиентской базой:

- Модуль регистрации новых клиентов и обработки запросов на доступ к тренировкам.

Описание основных модулей системы:

Модуль управления клиентами:

- Регистрация новых клиентов с детальной информацией, включая контактные данные.
- Возможность редактирования данных существующих клиентов и добавление новых записей.

- Отслеживание текущего статуса клиента, такого как активность, членство.

Модуль управления тренировочными программами:

- Создание новых тренировочных программ с указанием целей, типов тренировок.
- Редактирование существующих программ, добавление новых упражнений, изменение параметров тренировок.

Модуль управления членствами:

- Создание новых видов членств с указанием длительности, доступных тренировок и стоимости.
- Редактирование существующих членств, изменение условий, цены и сроков действия.
- Мониторинг статуса членства каждого клиента.

Модуль управления расписанием тренировок:

- Составление расписания тренировок с указанием временных слотов, тренеров и видов тренировок.
- Контроль доступа к определенным тренировкам в зависимости от типа членства.

Модуль администрирования:

- Управление персоналом центра, включая роли, права доступа и распределение обязанностей.
- Анализ статистики, включая отчеты о посещаемости, прибыли и эффективности программ.
- Мониторинг общего функционирования системы и ее производительности.

Эти модули обеспечивают глубокое взаимодействие и поддерживают разнообразные аспекты управления фитнес-центром "EpicPower", от ведения базы клиентов и создания тренировочных программ до управления членствами и анализа работы центра через административный модуль.

Код серверной части приложения:

```
urlpatterns = [
    path('home', home, name='home'),
    path('trainers', views.coach, name='trainers'),
    path('', home, name='home'),
    path('coach', coach, name='coach'),
    path('register', register, name='register'),
    path('login', handlelogin, name='auth'),
    path('logout', handlelogout, name='exit'),
    path('profile', profile, name='profile'),
    path('update', update, name='update'),
    path('stat/', stat, name='stat'),
    path('team_members', team_members, name='team_members'),
    path('price/', views.price_view, name='price'),
    path('schedule/', views.schedule_view, name='schedule'),
    path('update_profile/', update_profile, name='update_profile'),
    path('card/', membership_view, name='membership_view'),
]
```

Рисунок 27 – модуль urls

```
def home(request):
    return render(request, template_name: 'main/index.html')
```

Рисунок 28 – контроллер главной страницы

```

def register(request):
    if request.method == "POST":
        username = request.POST.get('login')
        email = request.POST.get('email')
        pass1 = request.POST.get('pass1')
        pass2 = request.POST.get('pass2')

        if pass1 != pass2:
            messages.info(request, message: "Пароли не совпадают")
            return redirect('/register')

        try:
            if User.objects.get(username=username):
                messages.warning(request, message: "Логин занят")
                return redirect('/register')

        except Exception as identifier:
            pass

        myuser = User.objects.create_user(username, email, pass1)
        myuser.save()
        messages.success(request, message: "Пользователь создан")
        return redirect('/login')

    return render(request, template_name: "main/register.html")

```

Рисунок 29 – контроллер страницы регистрации

```

def handlelogin(request):
    if request.method == "POST":
        username = request.POST.get('username')
        pass1 = request.POST.get('pass1')
        myuser = authenticate(username=username, password=pass1)
        if myuser is not None:
            login(request, myuser)
            messages.success(request, message: "Вы успешно вошли")
            return redirect('/')
        else:
            messages.error(request, message: "Неправильные данные")
            return redirect('/login')

    return render(request, template_name: "main/login.html")

```

Рисунок 30 – контроллер страницы авторизации

```

def handlelogout(request):
    logout(request)
    messages.success(request, message: "Вы успешно вышли")
    return redirect('/')

```

Рисунок 31 – контроллер выхода из аккаунта


```

def update(request):
    if request.method == 'POST':
        user_form = UpdateUserForm(request.POST, instance=request.user)
        profile_form = UpdateProfileForm(request.POST, instance=request.user.profile)
        if user_form.is_valid() and profile_form.is_valid():
            user_form.save()
            profile_form.save()
            messages.success(request, message: 'Ваш профиль был успешно обновлен!')
            return redirect('/profile')
        else:
            messages.error(request, message: 'Пожалуйста, исправьте ошибки.')
    else:
        user_form = UpdateUserForm(instance=request.user)
        profile_form = UpdateProfileForm(instance=request.user.profile)
    return render(request, template_name: 'main/update.html', context: {
        'user_form': user_form,
        'profile_form': profile_form
    })

```

Рисунок 32 – контроллер страницы изменения данных

```

def update_profile(request):
    if request.method == 'POST':
        selected_membership_id = request.POST.get('membership')
        selected_plan_id = request.POST.get('plan')

        # Получите объект выбранного членства и плана
        selected_membership = GymMembership.objects.get(pk=selected_membership_id)
        selected_plan = SubPlan.objects.get(pk=selected_plan_id)

        # Обновите профиль пользователя
        request.user.profile.selected_membership = selected_membership
        request.user.profile.selected_plan = selected_plan
        request.user.profile.save()

    return redirect('profile')

```

Рисунок 33 – контроллер страницы изменения членства

```

def membership_view(request):
    if request.user.is_authenticated:
        memberships = GymMembership.objects.all()
        plans = SubPlan.objects.all()
        context = {
            'memberships': memberships,
            'plans': plans,
        }
    if request.method == 'POST':
        selected_membership_id = request.POST.get('membership', None)
        selected_plan_id = request.POST.get('plan', None)
        if selected_membership_id and selected_plan_id:
            selected_membership = GymMembership.objects.get(pk=selected_membership_id)
            selected_plan = SubPlan.objects.get(pk=selected_plan_id)
            # Создаем или обновляем профиль пользователя с информацией о выбранном членстве и плане
            profile, created = Profile.objects.get_or_create(user=request.user)
            profile.selected_membership = selected_membership
            profile.selected_plan = selected_plan
            profile.save()
            messages.success(request, message: f'Абонемент "{selected_membership.name}" и план членства выбраны успешно!')
            return redirect('/profile')
        else:
            messages.error(request, message: 'Не удалось выбрать абонемент или план. Пожалуйста, попробуйте еще раз.')
    return render(request, template_name: 'main/card.html', context)
else:
    return redirect('/login')

```

Рисунок 34 – контроллер страницы членства

```

def stat(request):
    # Получение всех профилей
    profiles = Profile.objects.all()
    # Сбор уникальных членств и их подсчет в профилях
    membership_counts = {}
    for profile in profiles:
        membership_name = profile.selected_membership.name if profile.selected_membership else 'Нет членства'
        membership_counts[membership_name] = membership_counts.get(membership_name, 0) + 1
    # Сбор уникальных длительностей абонемента и их подсчет в профилях
    plan_counts = {}
    for profile in profiles:
        plan_months = profile.selected_plan.months if profile.selected_plan else 'Нет членства'
        plan_counts[plan_months] = plan_counts.get(plan_months, 0) + 1
    # Разделение данных для передачи в шаблон
    labels_membership = list(membership_counts.keys())
    data_membership = list(membership_counts.values())

    labels_plan = list(plan_counts.keys())
    data_plan = list(plan_counts.values())
    context = {
        'labels_membership': labels_membership,
        'data_membership': data_membership,
        'labels_plan': labels_plan,
        'data_plan': data_plan,
    }
    return render(request, template_name: 'main/stat.html', context)

```

Рисунок 35 – контроллер страницы со статистикой членств

```
def coach(request):
    coaches = Coach.objects.all()[:10]
    tren = Coach.objects.all()[:10]

    context = {"coaches": coaches, "tren": tren}
    return render(request, template_name: 'main/trainers.html', context)
```

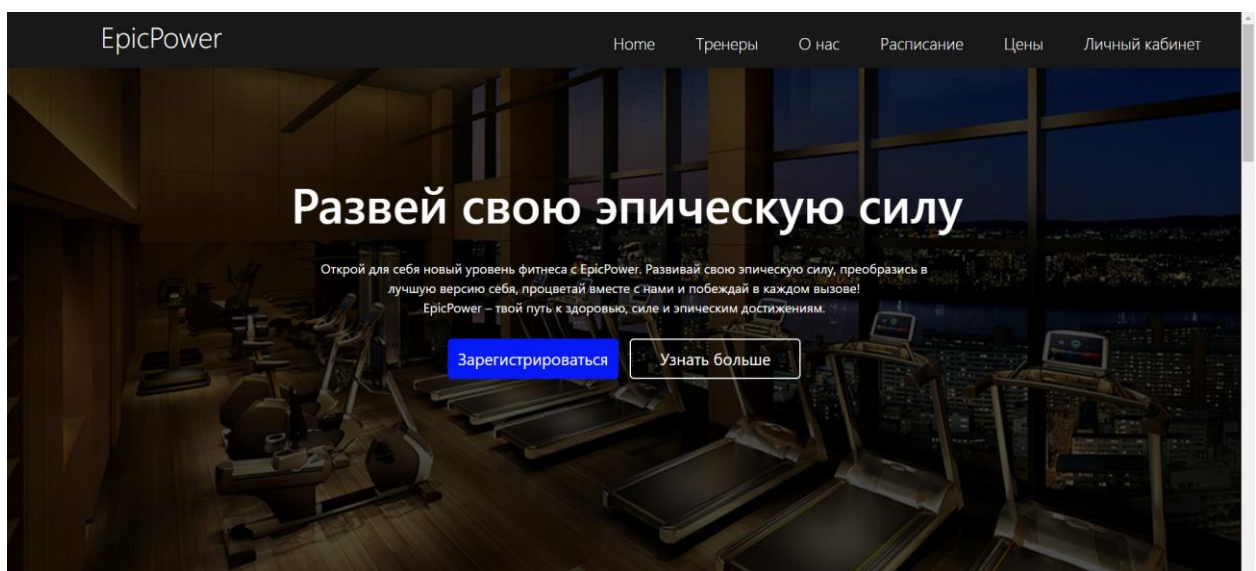
Рисунок 36 – контроллер страницы тренеров

```
def price_view(request):
    return render(request, template_name: 'main/price.html')

1 usage
def schedule_view(request):
    schedules = Schedule.objects.all()
    return render(request, template_name: 'main/schedule.html', context: {'schedules': schedules})
```

Рисунок 37 – контроллер страницы цен и расписания

Интерфейс пользователя



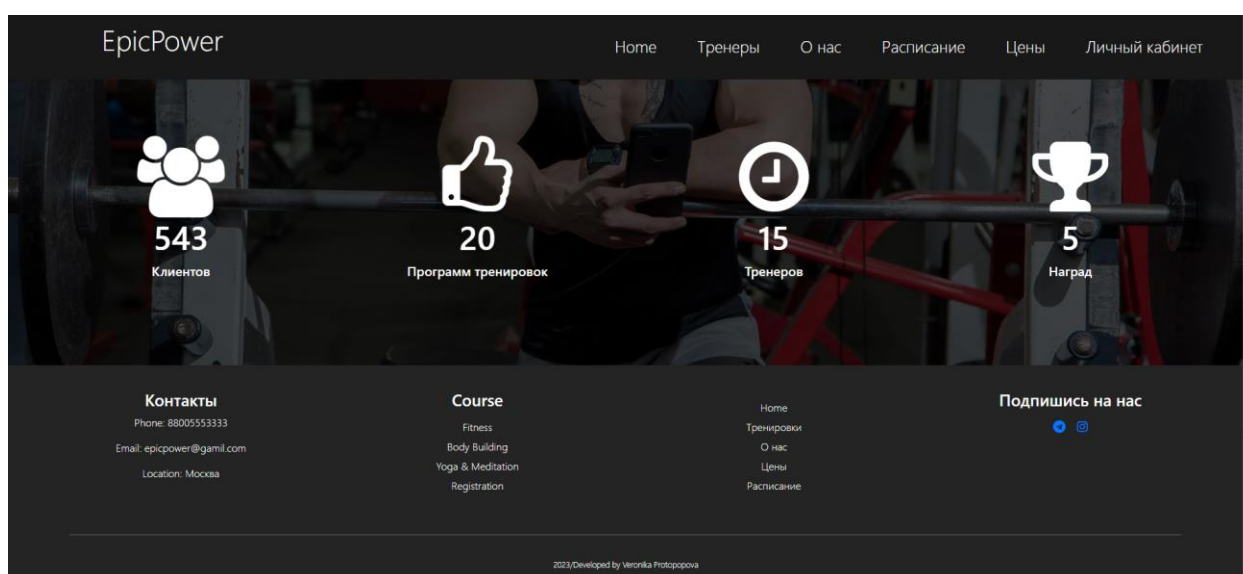
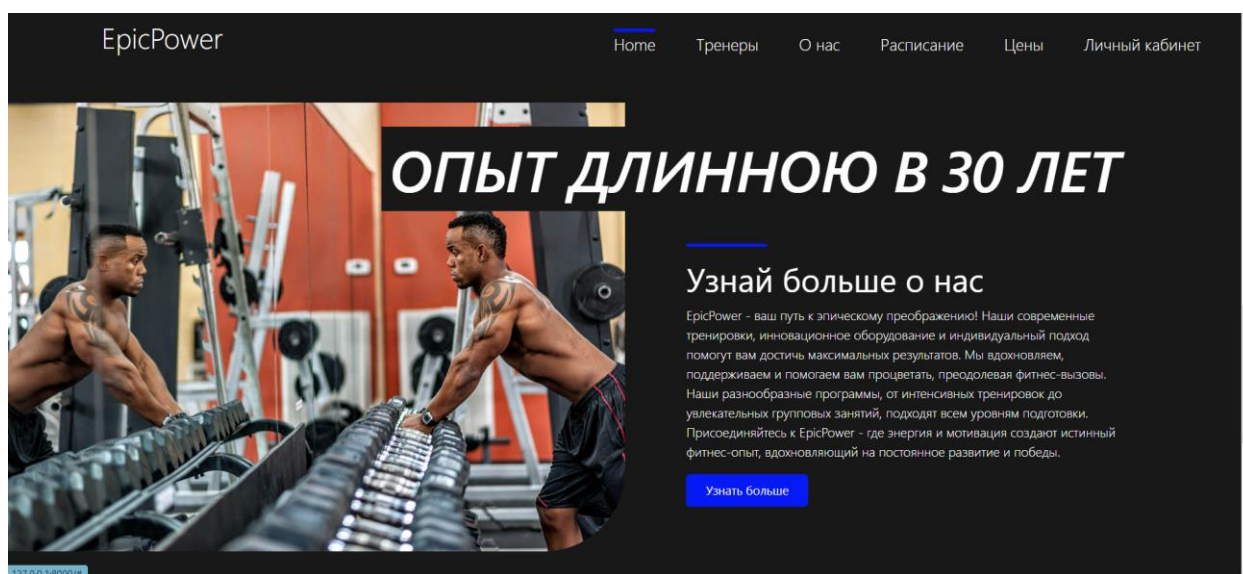
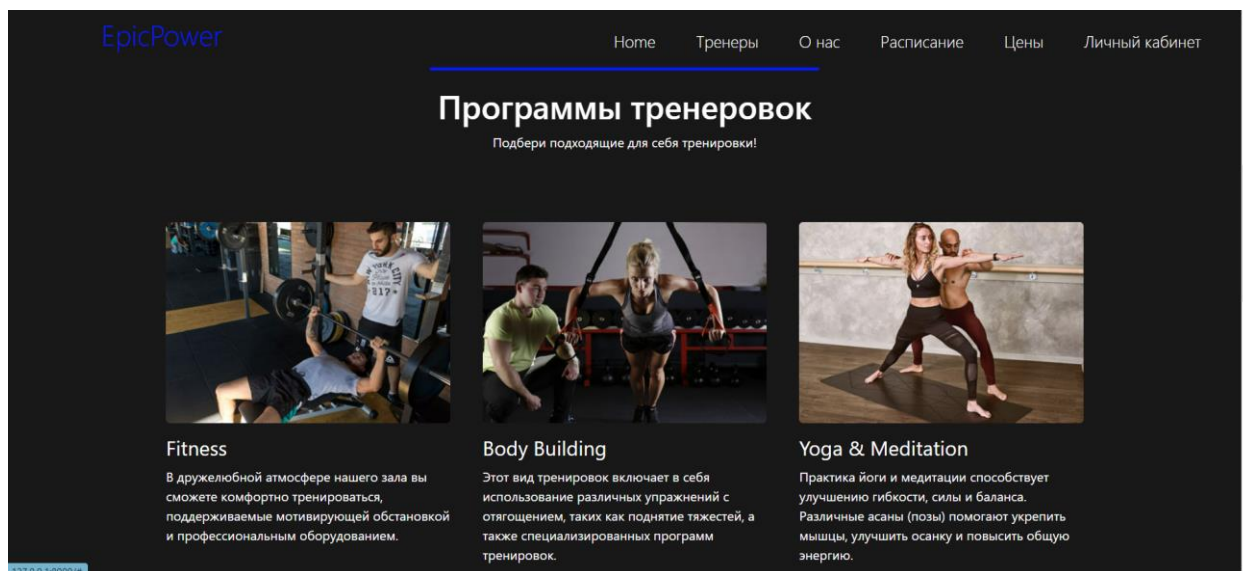


Рисунок 38 – Главная страница

В верхней части главной веб-страницы фитнес-центра "EpicPower" размещаются ключевые компоненты, включая логотип центра, его название и навигационные элементы для удобного перемещения по разделам сайта. Для посетителей без авторизации кнопка "Личный кабинет" направляет на страницу входа, а для зарегистрированных пользователей – в их личный кабинет, где также предоставлена опция "Выйти".

Главная страница содержит следующие разделы:

- О нас
- Программы тренировок
- Расписание (доступ к расписанию тренировок и мероприятий)
- Контакты (информация для связи)

Такая структура веб-сайта обеспечивает удобство посетителям в поиске информации и позволяет ознакомиться с разнообразными предложениями фитнес-центра "EpicPower".

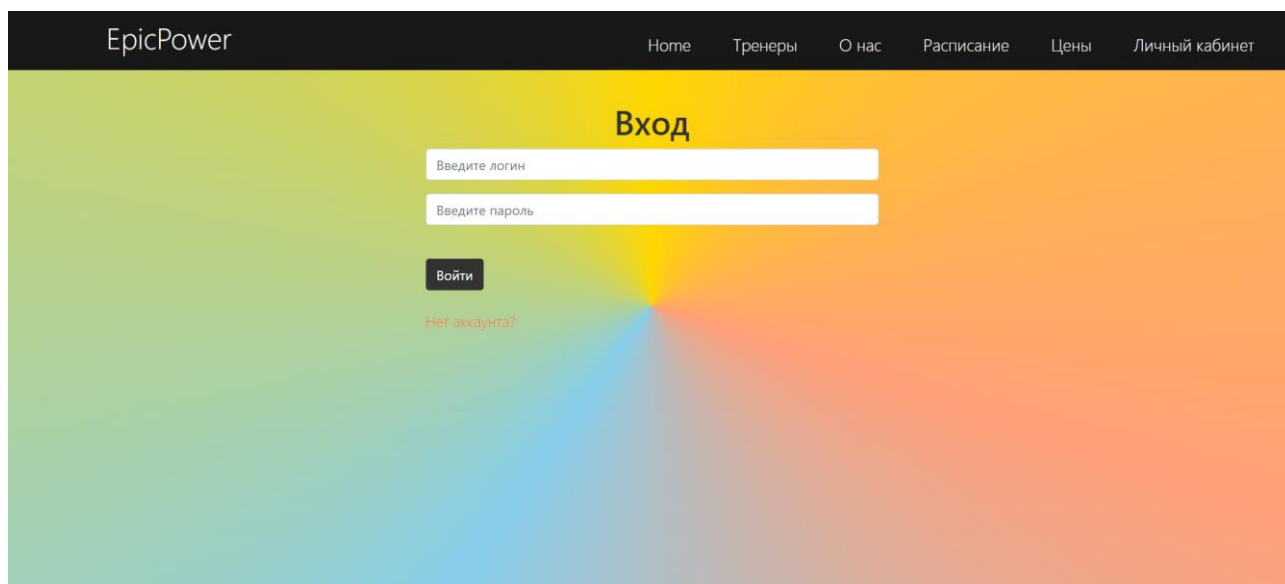


Рисунок 39 – Страница входа

EpicPower

Home

Тренеры

О нас

Расписание

Цены

Личный кабинет

Регистрация

Введите логин

Введите почту

Введите пароль

Подтвердите пароль

Зарегистрироваться

Есть аккаунт?

Рисунок 40 – Страница регистрации

EpicPower

Home

Тренеры

О нас

Расписание

Цены

Личный кабинет

Виды членства

Выбери себе членство, которое подойдёт по длительности и по количеству доступных программ.

Basic

2000 руб/Месяц

+ до 3 дней в неделю

+ доступ к групповым тренировкам

+ доступ к общему залу

+ доступ к аппарату InBody

Оформить

Standard

2999 руб/Месяц

+ до 5 дней в неделю

+ доступ ко всем оборудованям

+ доступ к приложению с онлайн-тренировками

+ доступ к аппарату InBody

Оформить

Premium

3899 руб/Месяц

+ неограниченное количество посещений

+ возможность индивидуальных занятий

+ доступ к приложению с онлайн-тренировками

+ доступ к аппарату InBody

Оформить

Оформление абонеента

Standart

3 месяцев

8997 Рублей

Оформить

Рисунок 41 – Страница для оформления членства клуба

При оформлении членства, есть возможность выбрать какой вид членства будет и в течение какого периода он будет действовать. Также в

зависимости от членства и длительности автоматически будет подсчитана его ИТОГОВАЯ СТОИМОСТЬ.

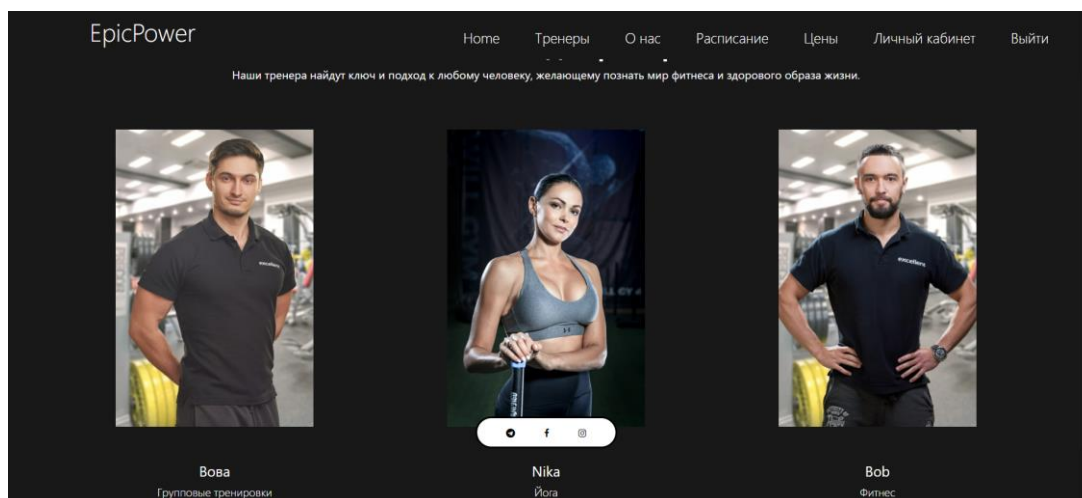


Рисунок 42 – Страница со списком тренеров

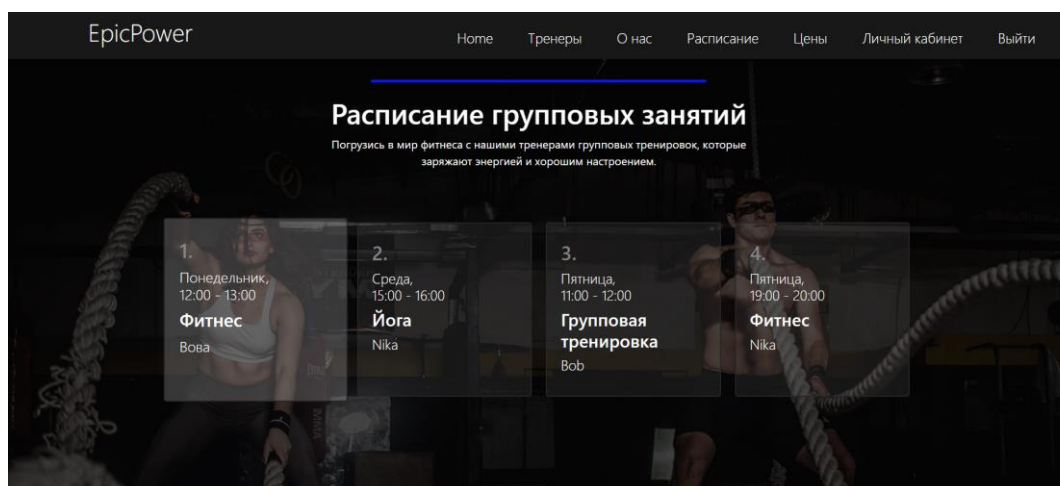



Рисунок 43 – Страница с расписанием



Логин : vero

Почта : vero@mail.ru

Имя : Вероника

Возраст : 22

Номер телефона : 89168300720

[Редактировать профиль](#)

Данные о членстве

Выбранное членство: Basic

Выбранный план: 6 месяцев

Количество посещений в неделю: 3

Доступ к приложению с онлайн-тренировками: Нет

Особенности членства: + доступ к аппарату InBody

Рисунок 44 – Личный кабинет

EpicPower

[Home](#)
[Тре](#)

Изменить данные

Рисунок 45 – Страница редактирования данных в личном кабинете

Отчётные данные



Рисунок 46 – Страница со статистикой для сотрудника

Эта страница доступна только для сотрудников, имеющих права администратора. Здесь отображается статистика по купленным видам членств и по их длительности.

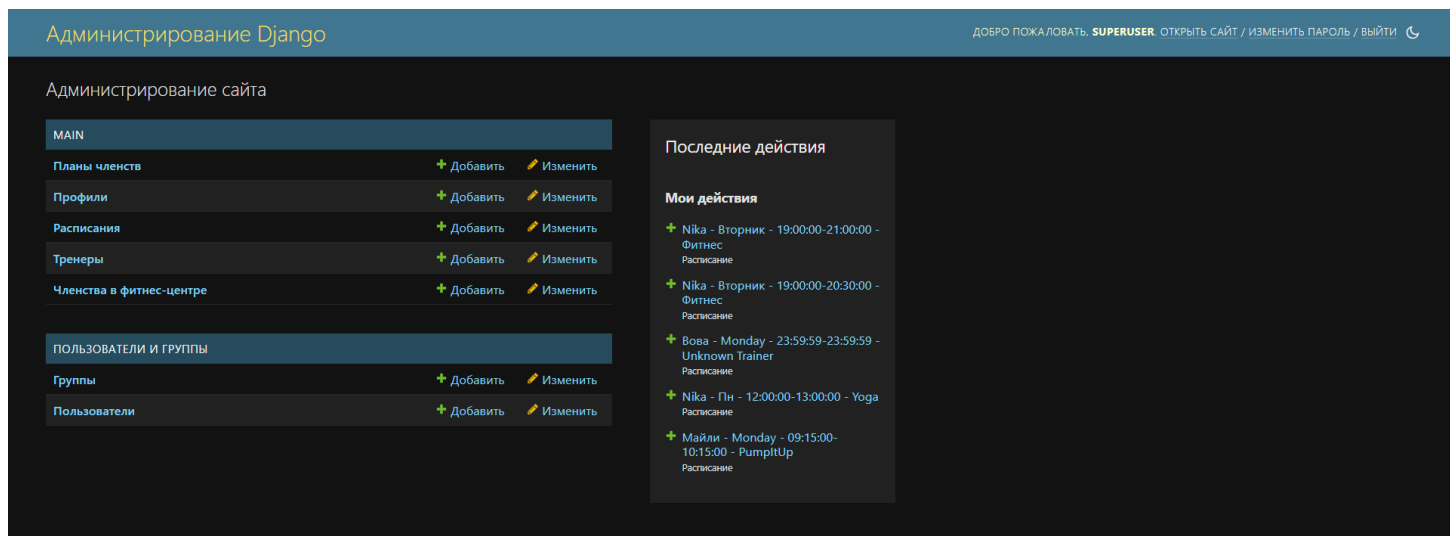


Рисунок 47 – Страница администратора.

Технологическая часть

Анализ исходных данных

Таблица 10 – Анализ исходных данных

Набор исходных данных	Проверяемый режим, функция, алгоритм
Не все обязательные поля в форме регистрации заполнены	Регистрация произведена не будет, выдаст ошибку «Заполните все поля»
Неверные данные пользователя	Выдаст ошибку «Введены неверные данные»
Отображение списка тренеров	Правильно представлена информация о тренерах со всеми данными
Отображение расписания	Правильно представлено расписание со всеми подробностями
Отображение главной страницы	Все данные корректно отображаются
При оформлении членства выбраны все необходимые поля	Корректно считается стоимость членства
Отображение личного кабинета	Если пользователь вошёл, то профиль отображается корректно со всеми данными и информацией о членстве.

Тестирующие драйверы и заглушки

Необходимо выполнить проверку функциональности в информационной системе по следующим аспектам:

- Процесс регистрации пользователя.
- Вход в учетную запись и отображение информации о пользователе.
- Навигация по различным страницам сайта.

- Оформление членства.
- Внесение изменений в данные пользователя.
- Проверка работы системы в различных веб-браузерах.

Пример драйвера для тестирования регистрации пользователя:

```
from django.contrib.auth.models import User
from django.urls import reverse
from django.test import TestCase

class RegistrationTest(TestCase):
    def test_registration_view(self):
        data = {
            'login': 'testuser',
            'email': 'testuser@example.com',
            'pass1': 'testpassword',
            'pass2': 'testpassword',
        }

        # Отправляем POST-запрос на URL для регистрации
        response = self.client.post(reverse('register'), data)
        # Проверяем, что после успешной регистрации происходит перенаправление
        self.assertEqual(response.status_code, 302)
        # Проверяем, что создан новый пользователь с указанным именем
        self.assertTrue(User.objects.filter(username='testuser').exists())
        # Проверяем, что произошло перенаправление на страницу входа
        self.assertRedirects(response, reverse('auth'))
        # Проверяем, что доступ к странице входа разрешен после регистрации
        response = self.client.get(reverse('auth'))
        self.assertEqual(response.status_code, 200) # Ожидаем успешный доступ
        # Проверяем, что используется правильный шаблон для страницы входа
        self.assertTemplateUsed(response, template_name='main/login.html')
```

Рисунок 48 – Пример драйвера тестирования авторизации

```
(venv) PS C:\Users\user\PycharmProjects\Fitness> python manage.py test main.tests
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
-----
Ran 1 test in 0.192s

OK
Destroying test database for alias 'default'...
```

Рисунок 49 – Результат тестирования авторизации

Протокол тестирования

Таблица 11 – Протокол тестирования

Входные данные	Ожидаемый результат	Полученный результат
Пользователь вводит корректный логин и пароль	Авторизация проходит успешно	Авторизация проходит успешно
Пользователь вводит некорректный логин	Ожидается сообщение об ошибке "Неверный логин"	Ожидается сообщение об ошибке "Неверный логин"
Пользователь оставляет поле пароля пустым	Ожидается сообщение об ошибке "Введите пароль"	Ожидается сообщение об ошибке "Введите пароль"
Работа навигации по сайту	При нажатии кнопки навигации пользователь корректно перенаправляется на соответствующую страницу.	При нажатии кнопки навигации пользователь корректно перенаправляется на соответствующую страницу.
Корректная работа сайта в разных браузерах	Все функции сайта выполняются правильно и дизайн web-приложения отображается корректно	Все функции сайта выполняются правильно и дизайн web-приложения отображается корректно

Исследование системы на производительность и отказоустойчивость

Современные веб-сайты сталкиваются с проблемой интенсивного потока запросов. Важно изучить реакцию системы на разные нагрузки – разное количество запросов в единицу времени. Для анализа использовался Apache Benchmark для стресс-тестирования и оценки реакции системы на высокую нагрузку. Представлены результаты исследования.

```
C:\Apache24\bin>ab -n 1000 -c 100 http://127.0.0.1:8000/
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      WSGIServer/0.2
Server Hostname:      127.0.0.1
Server Port:          8000

Document Path:        /
Document Length:      9403 bytes

Concurrency Level:     100
Time taken for tests:   44.772 seconds
Complete requests:      1000
Failed requests:         0
Total transferred:      9701000 bytes
HTML transferred:       9403000 bytes
Requests per second:    22.34 [#/sec] (mean)
Time per request:       4477.238 [ms] (mean)
Time per request:       44.772 [ms] (mean, across all concurrent requests)
Transfer rate:          211.60 [Kbytes/sec] received

Connection Times (ms)
              min    mean[+/-sd] median    max
Connect:        0    45 144.1      0      515
Processing:      6   4237 810.9    4626   4638
Waiting:        2   2280 1211.1    2063   4632
Total:          7   4282 806.1    4627   5143

Percentage of the requests served within a certain time (ms)
 50%    4627
 66%    4629
 75%    4630
 80%    4631
 90%    4633
 95%    4635
 98%    4637
 99%    4637
100%   5143 (longest request)
```

Рисунок 50 – Результаты тестирования

Исходя из данных стресс-тестирования, время обработки одного запроса в 4 мс свидетельствует о выдающейся эффективности системы. Это свидетельство того, что система способна обеспечивать высокую скорость обработки запросов, что является критически важным для обеспечения отзывчивости веб-приложения.

Полученные выводы из данных также подтверждают, что система проявляет отличную отказоустойчивость даже при высоких объемах запросов. Система продолжает функционировать стабильно и корректно даже при существенной нагрузке.

Эти результаты подчеркивают не только способность системы к быстрому реагированию на запросы, но и ее высокую степень устойчивости в условиях интенсивной активности. Такие характеристики являются ключевыми при обеспечении эффективности и надежности веб-приложения в различных сценариях использования.

ЗАКЛЮЧЕНИЕ

В процессе выполнения проекта, направленного на разработку информационной системы для фитнес-центра "EpicPower", сосредоточено внимание на тщательном анализе и разработке системы, начиная с формирования глоссария и концепции, и заканчивая созданием концептуальной модели предметной области и проектированием пользовательского интерфейса.

Основное внимание уделено не только структуре базы данных, но и другим важным компонентам проекта, чтобы обеспечить полноту и эффективность разработанной системы. Применение современных технологий, таких как Django, Python, HTML и CSS, позволило получить ценный опыт в области веб-разработки и информационных технологий.

Знания и навыки, полученные в ходе разработки информационной системы для фитнес-центра, представляют собой важные активы, готовые быть успешно применены в профессиональной сфере. В условиях постоянного развития фитнес-индустрии и внедрения новых технологий, эти компетенции становятся ключевыми для обеспечения конкурентоспособности и эффективности в области информационных технологий.

Таким образом, проект не только расширил знания и навыки в области веб-разработки, но и подготовил к успешной реализации информационных систем в сфере фитнес-центров. Усилия направлены на обеспечение эффективного функционирования системы и ее соответствия современным стандартам в данной отрасли.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Боровик И. Г. Управление данными [Электрон. ресурс] : метод. указания к курсовому проектированию / Боровик И. Г. ; МГТУ им. Н. Э. Баумана. - М. : Изд-во МГТУ им. Н. Э. Баумана, 2009. - ФГУП "Информрегистр" №0320901015 (дата обращения 23.10.2023).
2. Мацяшек Л.А. Т. Анализ требований и проектирование систем: Разработка информационных систем с использованием UML. –М.: Издательский дом «Вильямс», 2002.-432 с (дата обращения 19.11.2023).
3. В.В. Кириллов. Основы проектирования реляционных баз данных. СУБД - Учебные пособия и обзоры. ЦИТ. — учебное пособие СПбИТМО, 2011 (дата обращения 28.11.2023).
4. Ревунков Г. И., Ковалева Н. А., Силантьева Е. Ю. Проектирование баз данных : учеб. пособие / Ревунков Г. И., Ковалева Н. А., Силантьева Е. Ю. ; МГТУ им. Н. Э. Баумана. - М. : Изд-во МГТУ им. Н. Э. Баумана, 2018. - 45 с. : ил. - Библиогр.: с. 19. - Режим доступа: <http://ebooks.bmstu.ru/catalog/254/book1681.html>. - ISBN 978-5-7038-4718-3 (дата обращения 11.12.2023).