МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

Факультет компьютерных технологий и прикладной математики

**Кафедра информационных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №8**

**ДОКУМЕНТАЦИЯ**

Работу выполнил А.А.Козин

(подпись)

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Направленность Программирование и информационные технологии

Руководитель

канд. пед. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Н.Ю. Добровольская

(подпись)

Краснодар

2023

Оглавление

[Информация об Объекте 4](#_Toc133262859)

[Бизнес-процесс «Продажа» 5](#_Toc133262860)

[ПО присутствующее на рынке 9](#_Toc133262861)

[Экспресс расписание 10](#_Toc133262862)

[Сравнение характеристик 11](#_Toc133262863)

[Основные задачи создания ИС 13](#_Toc133262864)

[Перечень задач 14](#_Toc133262865)

[Функциональный требования 15](#_Toc133262866)

[Диаграмма требований 16](#_Toc133262867)

[Требования к информационному обеспечению 17](#_Toc133262868)

[Требование к инструментальному программному обеспечению 18](#_Toc133262869)

[Нефункциональный требования 19](#_Toc133262870)

[Проектирование функциональной структуры ПС 20](#_Toc133262871)

[Углубленная декомпозиция 21](#_Toc133262872)

[Укрупление декомпозиции 24](#_Toc133262873)

[Абстракция подсистемы 25](#_Toc133262874)

[Классификация абстракций 26](#_Toc133262875)

[Диаграмма классов 27](#_Toc133262876)

[Диаграммы состояний 28](#_Toc133262877)

[Диаграмма деятельности 29](#_Toc133262878)

[Диаграммы последовательности 30](#_Toc133262879)

[Идентефикация сущностей инфологической модели 31](#_Toc133262880)

[Разработка логической модели данных 32](#_Toc133262881)

[Проектирование базы данных системы 33](#_Toc133262882)

[Сравнительный анализ 34](#_Toc133262883)

[Программная реализация 35](#_Toc133262884)

[Описание базы данных 38](#_Toc133262885)

[Структура схема общесистемного программного обеспечения 42](#_Toc133262886)

[Перечень разработанных компонентов приложения 43](#_Toc133262887)

[Диаграмма компонентов 44](#_Toc133262888)

[Экранные формы приложения 45](#_Toc133262889)

[Анализ кода приложения по критерию сложности 48](#_Toc133262890)

[Физические элементы программной системы 49](#_Toc133262891)

[Ручное тестирование 50](#_Toc133262892)

[Нагрузочное тестирование 52](#_Toc133262893)

[Стрессовое тестирование 54](#_Toc133262894)

[Unit-тесты 55](#_Toc133262895)

# Информация об Объекте

Объект информатизации: компьютерные занятия «SwiftBook».

Организация была создана в 2017 году двумя разработчиками, которые увидели, что на Российском рынке нету курсов по языку разработки от компании Apple “Swift”. Занятия на курсе проходят в онлайн формате 2 раза в неделю в виде стрима. Курсы продаются через сайт. Объем годовых продаж: 8 потоков минимум 50 человек, 30 млн. руб.

Основная сфера деятельности: предоставление студентам основных и продвинутых знаний и навыков в области компьютерных технологий и информатики. Это может включать в себя обучение программированию, математике, управлению базами данных, сетевым технологиям, компьютерной графике, алгоритмам и структурам данных, кибербезопасности, цифровым маркетингом и т.д.

Цель - подготовить студентов к карьере в сфере информатики и технологий, а также повысить их компьютерную грамотность для личных и профессиональных целей.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 1 – направления развития бизнеса.

# Бизнес-процесс «Продажа»

Для информатизации выбирается процесс обработки заказа книг, запрошенных покупателем, подсчёт конечной стоимости, фиксация продажи в журнале отчёта и выставление счёта покупателю для оплаты.

К управляющей информации относится:

* Политика Конфиденциальности. Политика, которая запрещает распространять материалы курсов и личные данные аккаунта.
* Оферта на заключение договора возмездного оказания услуг. Важные условия договора, тип и объем услуг, цена и сроки.

К выходной информации относится:

* Формирование чека. Первичный документ, который применяется для оформления продажи товарно-материальных ценностей организации.
* Запись о подписанном договоре в журнал отчёта. Необходима для отслеживания деятельности и сборы статистики организации.

К действующим лицам данного бизнес-процесса относятся:

* Бухгалтер. Отвечает за ведение бухгалтерского учета и финансовых отчетов.
* Клиент. Обращается с заявкой на покупку курсов и предоставляет личные данные.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 2 – чёрный ящик бизнес-процесса.

Декомпозируем данный процесс для выявления задачи автоматизации.

Изображение выглядит как диаграмма, схематичный

Автоматически созданное описание

Рис. 3 – декомпозиция бизнес-процесса.

Согласно составленной декомпозиции, мы можем выявить следующие задачи:

* Задача формирования списка покупок;
* Задача формирования номера документа
* Задача указания данных об организации;
* Задача высчитывания общей стоимости.
* Задача формирования чека

Составим базовую модель вариантов использования:

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 4 – простая модель.

Опишем входные и выходные информационные потоки:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Наименование и назначение потока | Форма представле­ния | Обработчик | Корреспон­дент | Характеристики обработки | |
| Трудозатраты, чел’ч | Периодичность, регламент |
| 1 | Информация об имеющемся товаре | Электронная | Бухгалтер | База данных | 0,016 | По запросу |
|
| 2 | Заявка на покупку | Список  (электронный) | Бухгалтер | Клиент | 0,083 | По запросу |
|
| 3 | Данные о клиенте | Документ  (электронный) | Бухгалтер | Клиент | 0,083 | По запросу |
|

Таб. 1 – реестр входных информационных потоков

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Наименование и назначение потока | Форма представле­ния | Обработчик | Корреспон­дент | Характеристики обработки | |
| Трудозатраты, челч | Периодичность, регламент |
| 1 | Чек | Документ  (электронный) | Бухгалтер | Клиент | 0,016 | По запросу |
|
| 2 | Запись о подписанном договоре в журнал отчёта | Электронная | Бухгалтер | База данных | 0,016 | По запросу |
|

Таб. 2 - реестр выходных информационных потоков

Введём некоторые ограничения, которые будут необходимы для будущей реализации:

* Данные клиента(список) идентифицируются как уникальная запись в базе;
* Для начала оформления продажи товаров по заказу кассиру необходимо сгенерировать уникальный номер документа;
* Данные организации и информация о товаре заранее введены в базу и всегда известны.

Перечислим нормативно-справочную документацию, регламентирующую бизнес-процесс:

* Договор с правообладателем;
* Защита прав потребителей;
* Гражданский кодекс РФ;
* Документы регистрации юридического лица.

# ПО присутствующее на рынке

Для сравнения возьмём одно ПО с высоким функционалом и стоимостью, и один узконаправленное обеспечение с низкой стоимостью, например: 1С: Автоматизированное составление расписания и Экспресс расписание.

1С: Автоматизированное составление расписания

Программа для составления расписания предназначена для решения задач автоматизированного составления учебных расписаний и оперативного управления помещениями.  С его помощью составлять расписание можно в автоматическом, ручном и смешанном режимах с учетом многих ограничений и условий. При этом можно построить как допустимое расписание, так и оптимизированное, в котором сокращено количество окон или количество используемых помещений.

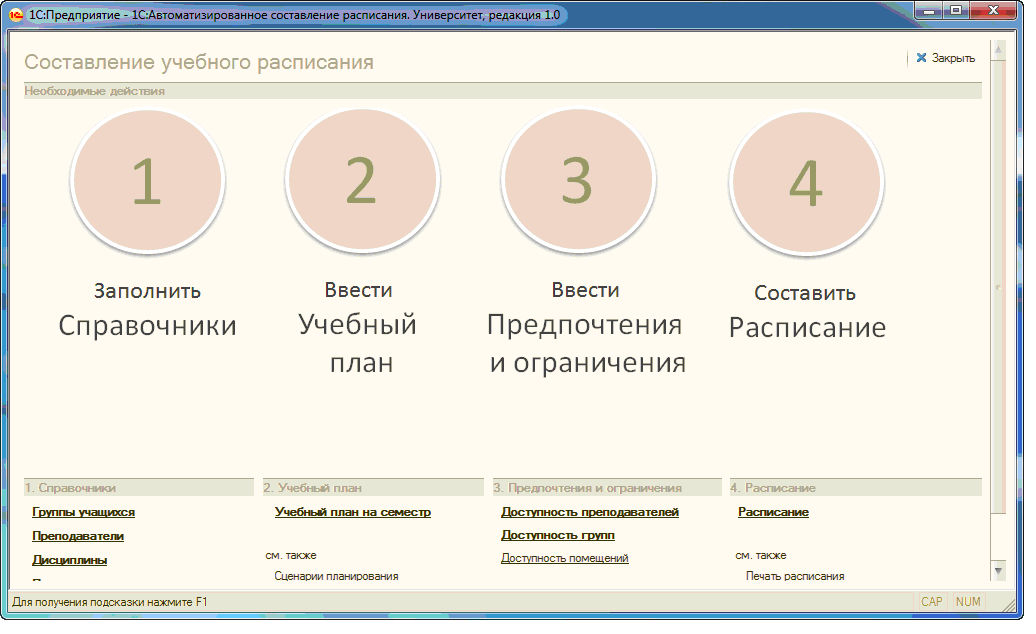


Рис. 5 – процесс составления расписания в 1С: Автоматизированное составление расписания

# Экспресс расписание

Программа "Экспресс-расписание" предназначена для автоматизации составления учебных занятий в колледжах, техникумах, училищах и профессиональных лицеях. Программа автоматически составляет основное расписание, позволяет вести ежедневные изменения расписания и учет выполненных часов, формирует отчеты.

Полная версия не имеет ограничений по функциональности программы и устанавливается на один рабочий компьютер. Будем рассматривать ее.

Изображение выглядит как стол

Автоматически созданное описание

Рис. 6 – планшет занятий

# Сравнение характеристик

Составим таблицу, отвечающую требованиям ГОСТ Р ИСО/МЭК 9126-93. Не будем включать в неё такие параметры как долговечность, безотказность, живучесть, т.к. эти термины подходят больше для системного обеспечения, а не программного.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Характеристика** | **1С: Автоматизированное составление расписания** | **Экспресс-расписание** |
| 1 | Безопасность | Информация хранится на сервере | Хранение информации на компьютере формирующего расписание, если другой человек знает, где находиться расписание, то он легко его может изменить |
| 2 | Стоимость | 26000 рублей | 7990 рублей |
| 3 | Наблюдаемость | Необходимо подключение доп. Сервисов 1С | Легко изменить расписание и вывести какую-то статистику, если человек имеет полную версию приложения |
| 4 | Сопровождаемость | Имеется горячая линия, но отсутствует постоянное обновление | Недостаточно информации |
| 5 | Наглядность | Простой и понятный интерфейс | Устаревший интерфейс, может кому-то быть не интуитивно понятным |
| 6 | Производительность | При использовании всего функционала может сильно нагружать систему на слабых компьютерах | При полной версии и использовании всего функционала практически нетребователен к системе. |
| 7 | Гибкость | Широкие возможности, для создания и отображения расписания и дальнейшей его автоматизации | Узкоспециализированное ПО, направленное на создание расписания. |
| 8 | Мобильность | Возможна работа только с другими продуктами 1С. | Не поддерживает работу с другими продуктами |
| 9 | Численность задействованного персонала | 1 человек | 1 человек |

Таб. 3 – сравнение характеристик

# Основные задачи создания ИС

Вид деятельности ИС: учёт, расчёт.

Перечень объектов автоматизации: кассовые аппараты, персональный компьютер, база данных.

Цели создания ИС

Автоматизация процесса оформления чека, автоматическое ведение учёта проданных компьютерных курсов, предоставление документации в электронном виде.

Критерий оценки достижения: созданная информационная система отображающая суть бизнес-процесса.

# Перечень задач

При выполнении данной работы планируется программным образом реализовать следующий перечень задач:

* Задача ввода списка компьютерных курсов и данных пользователя.
* Задача сохранения данных в базу данных.
* Задача формирования электронной накладной.

# Функциональный требования

* Требования к задаче «Задача ввода списка компьютерных курсов и данных пользователя»:

Система должна иметь возможность сопоставлять вводить данные в виде запроса к базе данных.

* Требования к задаче «Задача сохранения данных в базу данных»:

Все необходимые для работы системы данные должны находиться в базе данных. Должная иметься возможность удаления, добавления, редактирования и извлечения информации из базы данных.

* Требования к задаче «Задача формирования электронной накладной»:

Система должна иметь возможность сопоставлять вводимые данные с информацией в базе данных, формируя электронную накладную.

Электронная накладная составляется отдельно для каждого клиента, содержит перечень купленных им курсов, наименование организации и имя обслужившего кассира, согласно ГОСТ.

# Диаграмма требований

Расширим диаграмму UML с учётом предъявленных задач и требований:

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 7 – диаграмма требований

# Требования к информационному обеспечению

База данных информационной системы должна обладать следующими свойствами:

1. Надежность.  
   База данных должна быть надежной и обеспечивать целостность данных. Данные должны храниться в безопасном месте, чтобы избежать потери информации в результате сбоя системы или внешнего вмешательства.
2. Эффективность.  
   База данных должна обеспечивать высокую скорость доступа к данным и быструю обработку запросов. Это поможет ускорить процесс принятия решений и повысить эффективность работы системы в целом.
3. Масштабируемость.  
   База данных должна быть способной масштабироваться в соответствии с ростом объема информации и количества пользователей. Это поможет поддерживать эффективность работы системы в течение длительного времени.
4. Гибкость.  
   База данных должна обеспечивать гибкость в хранении и обработке данных, чтобы удовлетворять различным потребностям и требованиям пользователей.
5. Удобство.  
   Информационная система должна быть удобной в использовании, с простым интерфейсом и интуитивно понятными функциями. Это поможет упростить процесс работы с системой и повысить эффективность ее использования.
6. Безопасность.  
   База данных должна обеспечивать безопасность хранения и передачи данных, чтобы защитить конфиденциальную информацию от несанкционированного доступа.
7. Актуальность.  
   База данных должна содержать актуальную информацию о продажах компьютерных курсов и клиентах, чтобы обеспечить точность принятия решений на основе этих данных.

# Требование к инструментальному программному обеспечению

Необходимо обеспечить:

* Поддержку операционной системы Windows 10/11 и MacOS.
* Наличие пакета программ Microsoft Office
* Базу данных MySQL или MongoDB
* Поддержку пакетов Visual Studio и Visual Studio Code
* Поддержка сетевых протоколов TCP/IP.

# Нефункциональный требования

* Необходимо наличие интерфейса для работы.
* Должно быть постоянное питание системы.
* База данных для разрабатываемой системы не должна терять данные в случае случайного завершения системы.
* Наличие справки по созданному продукту для пользователя.

# Проектирование функциональной структуры ПС

Предоставим проектные решения по реализации функциональных требований к ПС. Для этого составим функциональную модель в нотации IDEF0 «Чёрный ящик», соответствующую требованиям, сформулированным при постановке задачи:

* Задача обработки данных клиента;
* Задача формирования списка покупок;
* Задача высчитывания общей стоимости;
* Задача формирования товарной накладной;
* Задача формирования номера документа;
* Задача добавления записи в журнал отчёта.

Договор с правообладателем

Законодательные документы для осуществления торговли

Товарная накладная

Заявка на покупку

Продажа курсов

Данные о клиенте

Информация   
о курсе

Запись о проведённой продаже в журнал отчёта

Клиент

Кассир

Рис. 8 – «Чёрный ящик бизнес-процесса»

# Углубленная декомпозиция

Декомпозиция является основным понятием стандарта IDEF0. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции.

Перед разработкой функциональной части ПС и обеспечивающих её подсистем необходимо дать более подробное представление о функционировании ПС. Раскроем их при помощи разукрупнённой модели чёрного ящика А0:



Рис.9 – Диаграмма декомпозиции А0.

Опишем элементы функциональной модели:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование стрелки** | **Источник стрелки** | **Тип стрелки источника** | **Приёмник стрелки** | **Тип стрелки приёмника** |
| Договор с правообладателем | Внешняя граница | Input | Получение информации об имеющихся курсе | Input |
| Законодательные документы для осуществления торговли | Внешняя граница | Input | Получение информации об имеющихся курсе | Input |
| Заяка на покупку | Внешняя граница | Input | Получение информации об имеющихся курсе | Input |
| Данные о клиенте | Внешняя граница | Input | Получение данных о клиенте | Input |
| Клиент | Внешняя граница | Input | Получение данных о клиенте | Input |
| Список покупок | Получение данных о клиенте | Output | Формирование конечной стоимости | Input |
| Передача паспортных данных | Получение данных о клиенте | Output | Формирование конечной стоимости | Input |
| Наличие потока курса | Получение информации об имеющемся товаре | Output | Формирование конечной стоимости | Input |
| Стоимость товара | Получение информации об имеющемся товаре | Output | Формирование конечной стоимости | Input |
| Данные о накладной заказа | Формирование конечной стоимости | Output | Учёт продажи | Input |
| Данные о статусе заказа | Формирование конечной стоимости | Output | Формирование накладной | Input |
| Запись о проведённой продаже в журнал отчёта | Учёт продажи | Output | Внешняя граница | Input |
| Товарная накладная | Формирование накладной | Output | Внешняя граница | Input |

Таб. 4 – описание модели.

# Укрупление декомпозиции

Раскроем несколько блоков декомпозиции А0 при помощи декомпозиции А2:



Рис.10 – декомпозиция А2 блока «Получение информации об имеющемся товаре»

Изображение выглядит как текст

Автоматически созданное описание

Рис. 11 – декомпозиция А2 блока «Формирование конечной стоимости»

# Абстракция подсистемы

Выявим набор абстракций предметной области проектируемой ПС. Разделим выделенные в проекте абстракции на три типа:

* Абстракции сущности;
* Абстракции поведения;
* Абстракции интерфейса.

Выделить возможное поведение каждой абстракции в пределах функциональности, проектируемой ИС, представленной моделью требований UML

|  |  |  |  |
| --- | --- | --- | --- |
| № | Абстракции | Тип | Описание |
| 1 | Данные о клиенте | Сущность | Паспортные данные |
| 2 | Кассир | Сущность | Пользователь системы, взаимодействующий с системой |
| 3 | Товарная накладная | Сущность | Список покупок, данные о клиенте, стоимость курса |
| 4 | Заявка о покупке | Сущность | Список покупок, ограничение по стоимости, дата |
| 5 | Запись о проведённой продаже в журнал отчёта | Поведение | Формирование записи в соответствии с проведённой операцией продажи и входных данных клиента. |

Таб. 5 – абстракция подсистемы.

# Классификация абстракций

Проведём классификацию обнаруженных абстракций по классическому подходу:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Класс | Тип | Описание |
| 1 | Человек | Сущность | Кассир |
| 2 | Предметы | Поведение | Данные о клиенте, заявка на покупку, товарная накладная. |
| 3 | События | Сущность | Запись о проведённой продаже в журнал, формирование накладной, обработка заявки |

Таб. 6 – классификация абстракций подсистемы.

# Диаграмма классов

Диаграмма классов — структурная диаграмма языка моделирования UML, которая демонстрирует общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними.

Эта диаграмма дает обзор программной системы путем отображения классов, атрибутов, операций и взаимосвязей.

Построим диаграмму классов UML.

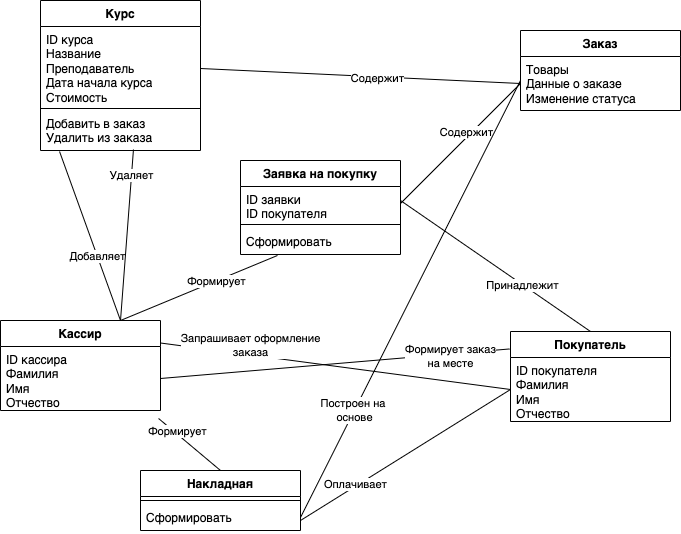


Рис. 12 – диаграмма классов.

# Диаграммы состояний

Диаграмма состояний — это способ моделирования поведения системы, который показывает, как система может находиться в различных состояниях в зависимости от различных входных событий и условий. Она является частью языка моделирования UML, который широко используется для разработки программного обеспечения.

Диаграмма состояний представляет систему в виде набора состояний, переходов между ними и действий, которые выполняются при переходе между состояниями.

Самые изменяемые классы из построенной диаграммы классов являются «Заказ» и «Накладная», т. к. зависят от множества других факторов.

Построим диаграмму состояний на основе выбранных классов.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 13 – диаграммы состояний.

# Диаграмма деятельности

Диаграмма деятельности — это графическое представление процесса или системы, которое отображает последовательность действий, которые должны быть выполнены, чтобы достичь определенной цели. Она используется в моделировании бизнес-процессов, программном проектировании и других областях.

Диаграмма деятельности позволяет визуализировать действия, переходы и условия, связанные с выполнением процесса или системы. На диаграмме деятельности могут быть представлены как последовательные шаги, так и параллельные процессы. Она может использоваться для анализа, проектирования и документирования бизнес-процессов, а также для описания функциональных требований к программному обеспечению.

Построим диаграмму деятельности.

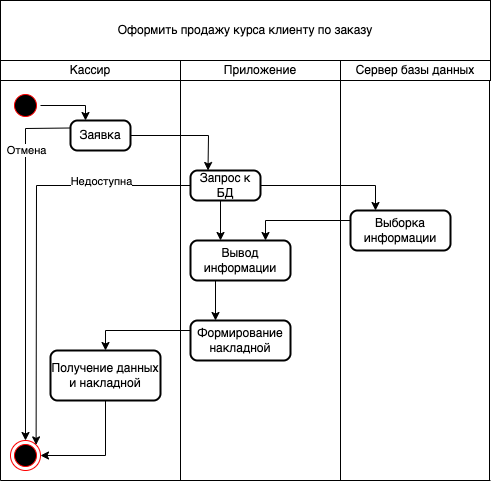


Рис. 14 – диаграмма деятельности.

# Диаграммы последовательности

Диаграмма последовательности — это графическое представление взаимодействия объектов в системе, показывающее последовательность сообщений, которые объекты посылают друг другу в определенной ситуации или в рамках определенного сценария.

Диаграмма последовательности включает в себя вертикальные линии, которые представляют различные объекты в системе, а также стрелки, которые показывают порядок передачи сообщений между этими объектами. Каждый объект на диаграмме может иметь свою жизненную линию, на которой отображается его состояние во время выполнения операции. Она позволяет легко понять последовательность выполнения операций и выявить потенциальные проблемы в процессе взаимодействия объектов.

Для перечисленных прецедентов создадим диаграмму последовательности.

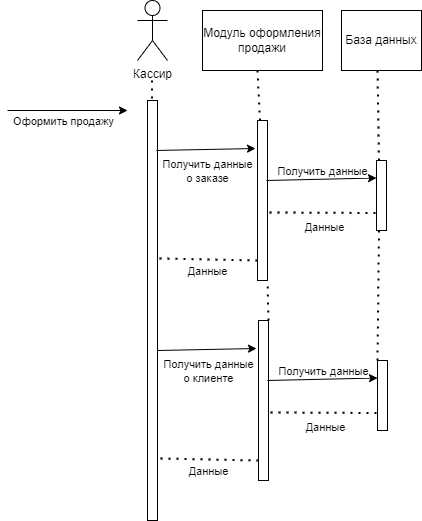


Рис. 15 – диаграмма последовательности.

# Идентефикация сущностей инфологической модели

На основе модели классов UML разработанной при выполнении лабораторно работы №4.1, выполним идентификацию сущностей информационной базы ПС.

|  |  |
| --- | --- |
| Документ | Сущность |
| Заявка для оформления курса | Заявка |
| Перечень курсов в заказе | Курсы в заказе |
| Профиль преподавателя в реестре | Преподаватель |
| Перечень курсов | Курсы |
| Запись о покупателе в журнале | Клиенты |

Таб. 7 – идентификация сущностей информационной базы ПС.

# Разработка логической модели данных

Логическая модель данных — это абстрактное представление данных, которое описывает структуру и отношения между различными объектами данных, независимо от физического хранения данных. Она служит промежуточным этапом между концептуальной моделью данных и физической моделью данных. Логическая модель данных описывает данные с точки зрения их использования и обработки в информационной системе.

Составим логическую модель данных:

|  |  |  |
| --- | --- | --- |
| Сущность | Атрибуты | Описание |
| Справочные | | |
| 1 Преподаватели | **ID преподавателя**, фамилия, имя, отчество, предмет | Информация о преподавателях |
| 2 Курсы | **ID курса**, наименование, программа обучения, стоимость | Информация о курсах |
| Оперативные | | |
| 3 Курсы в заказе | **ID связи**, *ID заявки, ID курса* | Информация о принадлежности курсов к заказу |
| 4 Заявки | **ID заявки**, *ID клиента* | Информация о заявках на клиента |
| 5 Клиенты | ID клиента, фамилия, имя, отчество, почта | Информация о клиентах |

Таб. 8 – сущности физической модели данных.

# Проектирование базы данных системы

Составим логическую модель проектируемой базы данных:

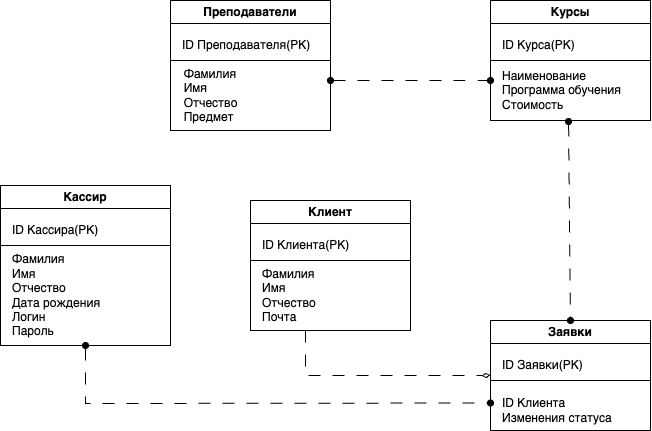


Рис. 16 – ER-диаграмма логической модели данных.

# Сравнительный анализ

Сравнительный анализ с диаграммой классов, изображённой на Рис.1 показал необходимость удаления сущности «Клиент», вследствие собственной избыточности. Так же необходимо добавить новую сущность «товары в заказе» для формирования списка, и «Статус заказа» для отслеживания этапов выполнения.

На основе выводов анализа составим физическую модель данных:

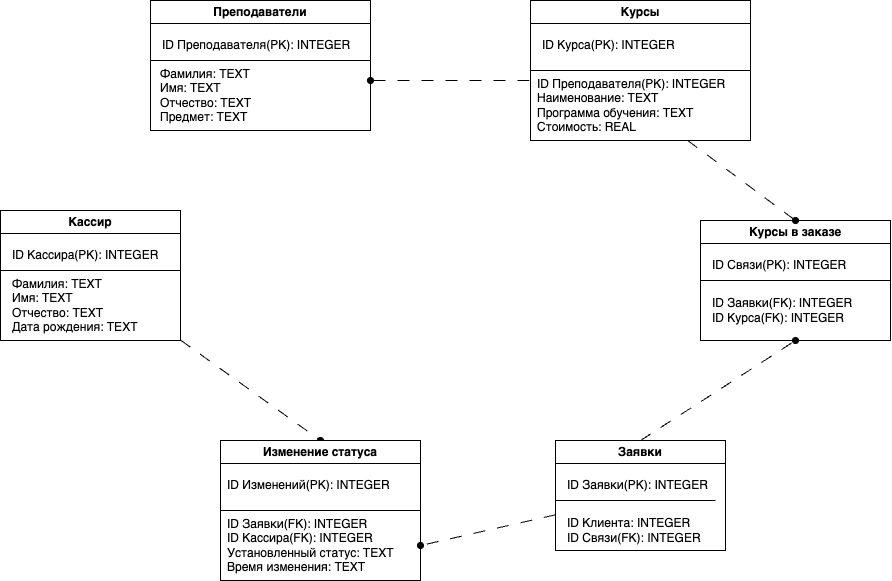


Рис. 17 – ER-диаграмма физической модели данных.

# Программная реализация

Для разработки ПС была выбрана СУБД PostgreSQL, по следующим причинам: бесплатное распространение, простота внедрения, поддержка БД неограниченного размера. В соответствии с построенной физической моделью данных, напишем SQL-скрипт:

CREATE TABLE IF NOT EXISTS teachers (

teacher\_id integer PRIMARY KEY,

first\_name text NOT NULL

second\_name text NOT NULL,

third\_name text,

subject text NOT NULL

);

CREATE TABLE IF NOT EXISTS courses (

courses\_id integer PRIMARY KEY,

teacher\_id integer NOT NULL,

courses\_name text NOT NULL,

program\_of\_study TEXT,

sale real NOT NULL,

FOREIGN KEY (cours\_id) REFERENCES courses (cours\_id)

);

CREATE TABLE IF NOT EXISTS product\_list (

connect\_id integer PRIMARY KEY,

request\_id integer NOT NULL,

cours\_id integer NOT NULL,

FOREIGN KEY (cours\_id) REFERENCES books (cours\_id),

FOREIGN KEY (request\_id) REFERENCES requests (request\_id)

);

CREATE TABLE IF NOT EXISTS requests (

request\_id integer PRIMARY KEY,

client\_id integer NOT NULL,

connect\_id integer NOT NULL,

FOREIGN KEY (connect\_id) REFERENCES product\_list (connect\_id)

);

CREATE TABLE IF NOT EXISTS status\_change (

change\_id integer PRIMARY KEY,

request\_id integer NOT NULL,

cashier\_id integer NOT NULL,

present\_status text NOT NULL,

time\_of\_change text NOT NULL,

FOREIGN KEY (request\_id) REFERENCES requests (request\_id),

FOREIGN KEY (cashier\_id) REFERENCES cashiers (cashier\_id)

);

CREATE TABLE IF NOT EXISTS cashiers (

cashiers\_id integer PRIMARY KEY,

first\_name text NOT NULL,

second\_name text NOT NULL,

third\_name text,

date\_of\_birth text NOT NULL

);

CREATE TABLE IF NOT EXISTS clients (

Id INTEGER PRIMARY KEY AUTOINCREMENT,

FirstName TEXT NOT NULL,

LastName TEXT NOT NULL,

PhoneNumber TEXT NOT NULL,

Email TEXT NOT NULL

);

# Описание базы данных

Составим структуру каждой включенной в схему таблицы:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Размер | Условие на значение | Значение по умолчанию | Примечания |
| teacher\_name | text | - | NOT NULL | - |  |
| first\_name | text | - | NOT NULL | - |  |
| second\_name | text | - | NOT NULL | - |  |
| third\_name | text | - |  | - |  |
| subject | text | - | NOT NULL | - |  |
| teacher\_id | integer | - | NOT NULL | - | PK |

Таб. 9 – структура таблицы teachers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Размер | Условие на значение | Значение по умолчанию | Примечания |
| cours\_id | integer | - | NOT NULL | - | PK |
| teacher\_id | integer | - | NOT NULL | - | FK |
| cours\_name | text | - | NOT NULL | - |  |
| program\_of\_study | text | - |  | - |  |
| sale | real | - | NOT NULL | - |  |

Таб. 10 – структура таблицы courses

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Размер | Условие на значение | Значение по умолчанию | Примечания |
| connect\_id | integer | - | NOT NULL | - | PK |
| request\_id | integer | - | NOT NULL | - | FK |
| cours\_id | integer | - | NOT NULL | - | FK |

Таб. 11 – структура таблицы product\_list

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Размер | Условие на значение | Значение по умолчанию | Примечания |
| request\_id | integer | - | NOT NULL | - | PK |
| client\_id | integer | - | NOT NULL | - |  |
| connect\_id | integer | - | NOT NULL | - | FK |

Таб. 12 – структура таблицы requests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Размер | Условие на значение | Значение по умолчанию | Примечания |
| change\_id | integer | - | NOT NULL | - | PK |
| request\_id | integer | - | NOT NULL | - | FK |
| cashier\_id | integer | - | NOT NULL | - | FK |
| present\_status | text | - | NOT NULL | - |  |
| time\_of\_change | text | - | NOT NULL | - |  |

Таб. 13 – структура таблицы status\_change

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Размер | Условие на значение | Значение по умолчанию | Примечания |
| cashier\_id | integer | - | NOT NULL | - | PK |
| first\_name | text | - | NOT NULL | - |  |
| second\_name | text | - | NOT NULL | - |  |
| third\_name | text | - |  | - |  |
| date\_of\_birth | text | - | NOT NULL | - |  |

Таб. 14 – структура таблицы cashier

|  |  |  |
| --- | --- | --- |
| № п/п | Имя таблицы | Описание таблицы |
| 1 | teachers | Справочная информация о преподавателях и их предметах, которые они ведут |
| 2 | courses | Перечень курсов |
| 3 | product\_list | Список курсов, которые составляет клиент |
| 4 | requests | Заявка, объединяющая клиента и его список |
| 5 | status\_change | Отслеживает состояние покупки |
| 6 | cassirs | Справочная информация об кассирах |

Таб. 15 – список разработанных таблиц

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Родительская таблица | | Дочерняя таблица | | Тип связи |
| Название | Атрибут | Название | Атрибут |
| teacher | teacher\_id | courses | teacher\_id | один-к-одному |
| courses | cours\_id | product\_list | cours\_id | один-к-одному |
| requests | request\_id | product\_list | request\_id | один-ко-многим |
| product\_list | connect\_id | requests | connect\_id | один-к-одному |
| requests | request\_id | status\_change | request\_id | один-ко-многим |
| cashier | cashier\_id | status\_change | cashier\_id | один-к-одному |

Таб. 16 – список связей между таблицами базы данных

# Структура схема общесистемного программного обеспечения

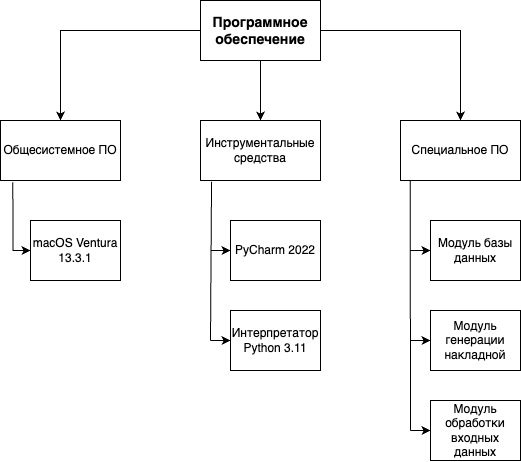


Рис. 18 – схема приложения.

# Перечень разработанных компонентов приложения

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя | Тип | Описание |
| 1 | main.py | source | Основной системный файл, через который реализуется большая логическая часть работы приложения, включая поддержку SQL запросов. |
| 2 | gui.py | source | Системный файл, отвечающий за отображение и функционирование оконной части. |
| 3 | create\_database.py | source | Системный файл с кодом, при запуске которого создается база данных |
| 4 | database\_export.xlsx | source | Файл, в который экспортируется база данных. |

Таб. 17 – перечень компонентов.

# Диаграмма компонентов

Изображение выглядит как письмо

Автоматически созданное описание

Рис. 19 – диаграмма компонентов приложения

# Экранные формы приложения

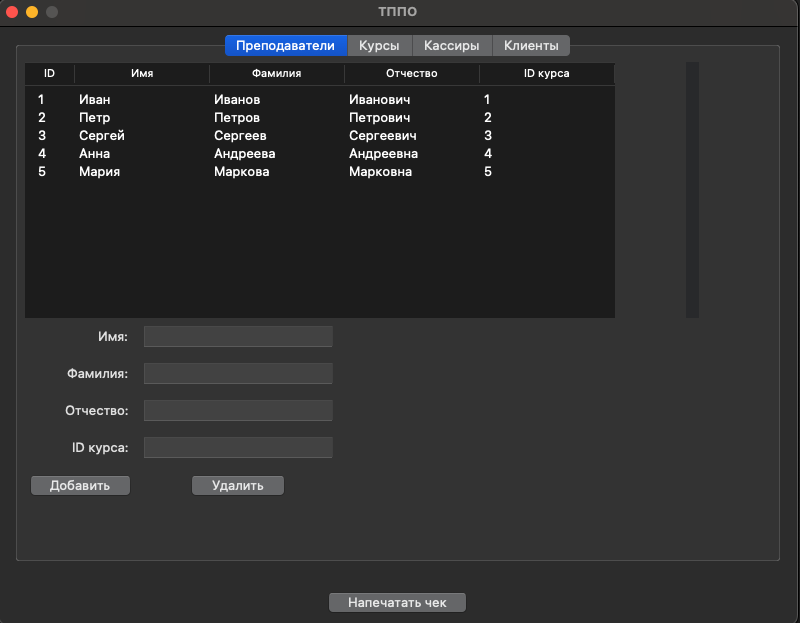


Рис. 20 – Вкладка “Преподаватели”

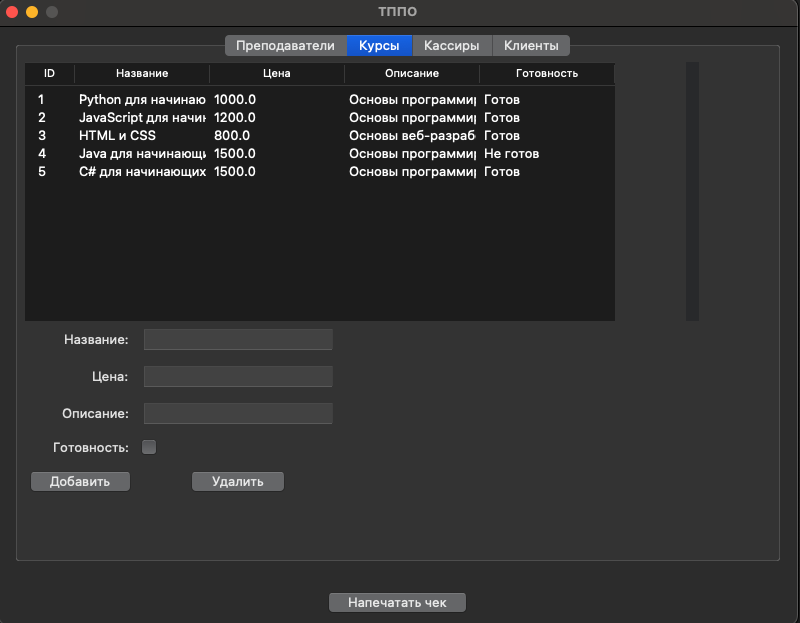


Рис. 21 – Вкладка “Курсы”

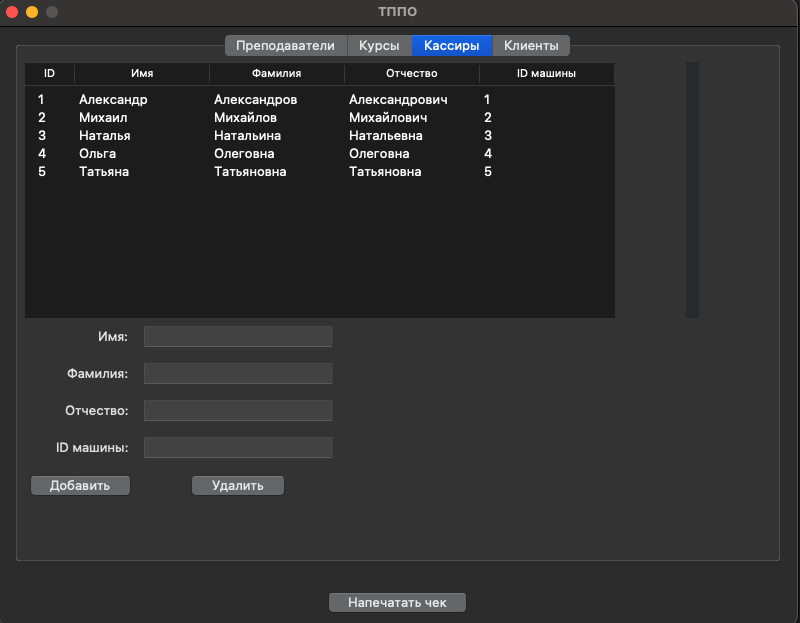


Рис. 22 – Вкладка “Кассиры”

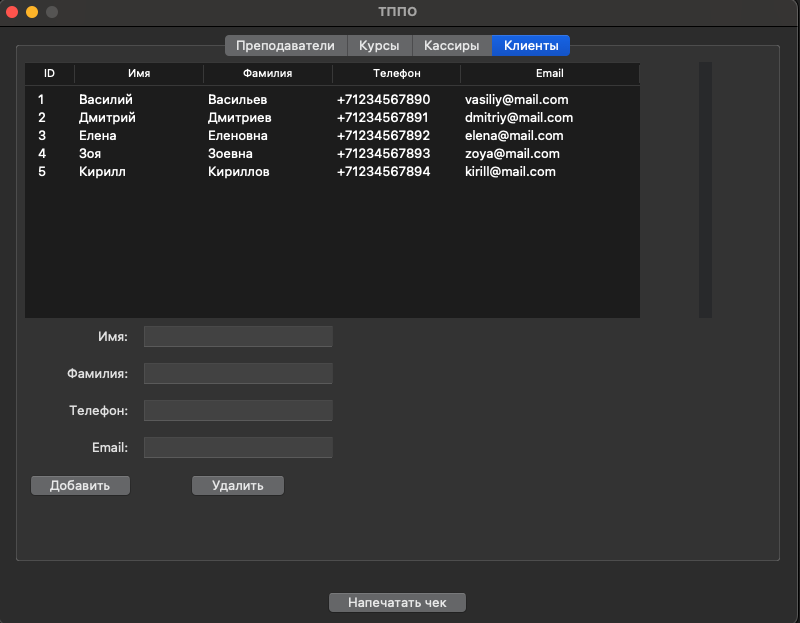


Рис. 23 – Вкладка “Клиенты”

Изображение выглядит как стол

Автоматически созданное описание

Рис. 24 – Итоговый вид вывода приложения

# Анализ кода приложения по критерию сложности

* Число модулей приложения – 26
* Суммарное число переменных подпрограмм, включая их формальные параметры – 96
* Суммарное количество операторов подпрограмм –24
* Глубина вложенности структурных операторов ветвления и повторения - 1

# Физические элементы программной системы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя | Тип | Описание |
| 1 | Рабочая станция | Процессор | Основная рабочая станция пользователя |

Таб. 18 – перечисление физических элементов ПС

# Ручное тестирование

Были протестированы следующие функции системы:

* Добавление и удаление преподавателей/курсов/кассиров/клиентов
* Добавление и удаление курсов
* Добавление и удаление кассиров
* Добавление и удаление клиентов
* Изменение списка преподавателей/курсов/кассиров/клиентов
* Экспорт базы данных в таблицу Microsoft Excel

В результате ручного тестирования была найдена следующая ошибка.

*Последовательность для обнаружения*: при удалении преподавателя/курса/кассира/клиента под номером N, следующая добавленная сущность будет под номером N+1, а не заменять удаленную сущность.

*Внешнее проявление ошибки* представлено на рисунке 1.

*Способ решения*: в классе преподавателя/курса/кассира/клиента переработать логику добавления и удаления. Также можно добавить кнопку “Изменить” для удобства пользователя.

Изображение выглядит как текст, монитор, снимок экрана, черный

Автоматически созданное описание

Рис. 25 – внешнее проявление ошибки.

# Нагрузочное тестирование

Технические характеристики машины:

* Процессор – Apple M1 Pro
* Оперативная память – 16ГБ
* Жесткий диск – Macintosh HD 512 ГБ

Для проведения нагрузочного тестирования используется библиотека “Faker”. Она позволяет генерировать большое количество данных, которые будут использованы для заполнения таблиц.

Было проведено нагрузочное тестирование на следующих объемах случайных данных:  
 N преподавателей, N курсов, N кассиров, N клиентов

Для значений N = 100, 500, 1000, 2500, 5000. Были произведены замеры времени экспорта данных в Excel и загрузки главного окна, в него входят:

* Список всех преподавателей
* Список всех курсов
* Список всех кассиров
* Список всех клиентов

По результатам нагрузочного тестирования, заметно меняется лишь время загрузки самого окна.

На таблице 1 и рисунке 2 приведены результаты по замерам.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Замеры | N = 100 | N = 500 | N = 1000 | N = 2500 | N = 5000 |
| Главное окно | 0.14 | 0.19 | 0.24 | 0.3 | 0.35 |
| Экспорт в Excel | 0.15 | 0.18 | 0.25 | 0.32 | 0.36 |

Таб. 19 – результаты нагрузочного тестирования (в секундах).

Рис. 26 – зависимость времени выполнения от N

# Стрессовое тестирование

Стрессовое тестирование проводилось на случайных данных, сгенерированных с N = 100000, после чего было замерено время ответа каждого из окон. Также был найден верхний порог генерации данных, они не генерируются начиная с N = 500000.

Результаты измерений приведены в таблице 2

|  |  |
| --- | --- |
| Измерения | N = 100000 |
| Главное окно | 2.5 |
| Экспорт в Excel | 15 |

Таб. 20 – результаты стрессового тестирования (в секундах).

Рис. 27 – график зависимости затраченного времени от количества записей при наличии большой сторонней нагрузки на оборудование

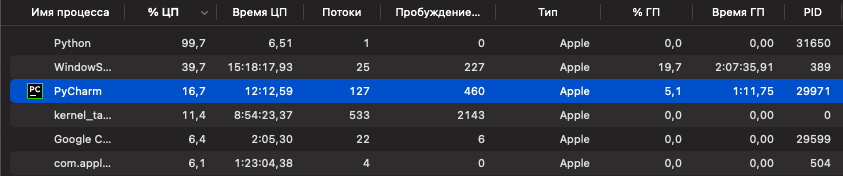


Рис. 28 – нагрузка ноутбука

# Unit-тесты

Пустой проект Unit-тестов:

**import** unittest  
**import** sqlite3  
**from** main **import** Application  
**from** tkinter **import** Tk  
**from** database **import** create\_tables, insert\_sample\_data  
  
  
**class** TestApp(unittest.TestCase):  
  
 **def** setUp(self):  
 # Создание базы данных для тестирования  
 self.conn = sqlite3.connect(":memory:")  
 create\_tables(self.conn)  
 insert\_sample\_data(self.conn)  
  
 self.root = Tk()  
 self.app = Application(self.root, self.conn)  
  
 **def** tearDown(self):  
 self.conn.close()  
 self.root.destroy()  
  
 **def** test\_instructors\_tab(self):  
 # Добавление нового преподавателя  
 self.app.instructors\_frame.instructor\_first\_name\_entry.insert(0, "Иван")  
 self.app.instructors\_frame.instructor\_last\_name\_entry.insert(0, "Иванов")  
 self.app.instructors\_frame.instructor\_middle\_name\_entry.insert(0, "Иванович")  
 self.app.instructors\_frame.add\_instructor\_button.invoke()  
  
 # Проверка, что новый преподаватель добавлен в список  
 last\_instructor = self.app.instructors\_frame.instructors\_listbox.get(  
 self.app.instructors\_frame.instructors\_listbox.size() - 1)  
 self.assertEqual(last\_instructor, "Иванов Иван Иванович")  
  
 # Удаление преподавателя  
 self.app.instructors\_frame.instructors\_listbox.selection\_set(0)  
 self.app.instructors\_frame.delete\_instructor\_button.invoke()  
  
 # Проверка, что преподаватель был удален  
 deleted\_instructor = self.app.instructors\_frame.instructors\_listbox.get(0)  
 self.assertNotEqual(deleted\_instructor, "Иванов Иван Иванович")  
  
  
**if** \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()