МИНИСТЕРСТВО ОБРАЗОВАНИЯ НИЖЕГОРОДСКОЙ ОБЛАСТИ Государственное бюджетное профессиональное образовательное учреждение НИЖЕГОРОДСКИЙ РАДИОТЕХНИЧЕСКИЙ КОЛЛЕДЖ

**КУРСОВАЯ РАБОТА**

**МДК 02.01 Информационные технологии и платформы разработки информационных систем**

**Тема: «Разработка информационной системы для настольного клуба»**

Выполнил студент Проверил преподаватель

группы 4ИСиП-19-1 Гутянская Е.М.

Танцев В. В. Проект защищен с оценкой

ФИО \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2022 г.

**ЗАДАНИЕ**

**на курсовую работу**

**МДК 02.01 Информационные технологии и платформы разработки информационных систем**

Студенту Танцеву Владиславу Владимировичу

Группа 4ИСиП-19-1

**Тема курсовой работы «*Разработка информационной системы для клуба настольных игр*»**

Дата выдачи задания « » 2022 г.

Срок сдачи работы « » 2022 г.

**Перечень вопросов, подлежащих разработке:**

- Введение (название выбранной темы, обзор раскрываемых вопросов).

- Разработка технического задания.

- Проектирование информационной системы.

- Разработка информационной системы.

- Тестирование информационной системы.

- Руководство пользователя.

- Заключение.

- Список использованной литературы.

**Перечень диаграмм проектирования:**

1. Диаграмма прецедентов. Сценарий варианта использования.
2. Информационно-логическая модель базы данных.

**Задание выдал преподаватель** /Е.М. Гутянская/

**Оглавление**

Введение………………………………………………………………………………………….5

1. Теоретические основы разрабатываемой темы……………………………………………...6

1.1. Анализ проектируемой системы……………………………………………………6

1.2. Обоснование выбора средств разработки информационной системы …………..6

2. Разработка технического задания…………………………………………………………….8

2.1. Назначение и цели создания системы………………………………………………8

2.1.1. Назначение системы……………………………………………………….8

2.1.2. Цели создания системы……………………………………………………8

2.2. Характеристика объектов автоматизации………………………………………….8

2.3. Требования к структуре системы…………………………………………………...8

2.3.1. Требования к структуре и функционирования системы…………………8

2.3.2. Требования к способам и средствам связи для информационного обмена между компонентами системы…………………………………………………..8

2.3.3. Требования к режимам функционирования системы……………………9

2.3.4. Требования по диагностированию системы……………………………..9

2.3.5. Требования к надежности…………………………………………………9

2.3.6. Требования к защите информации………………………………………..9

2.4. Требования к функциям, выполняемые системой………………………………..10

2.5. Требования к видам обеспечения………………………………………………….10

2.5.1. Требования к программному обеспечению……………………………..10

2.5.2. Требования к техническому обеспечению……………………………...10

3. Проектирование информационной системы………………………………………………..11

3.1. Диаграмма прецедентов (вариантов использования)…………………………….11

3.2. Диаграмма последовательности…………………………………………………...11

3.3. Информационно-логистическая модель базы данных…………………………...12

3.3.1. Описание структуры БД………………………………………………….12

4. Разработка информационной системы……………………………………………………...14

4.1. Описание структуры системы……………………………………………………..14

4.2. Разработка файла и подключения к базе данных…………………………………14

4.3. Разработка раздела «Авторизация пользователя»………………………………..16

4.4. Разработка раздела «Профиль сотрудника»………………………………………19

4.5. Разработка раздела «Добавление новой услуги»…………………………………26

4.6. Разработка раздела «Регистрация нового клиента»………………………………28

4.7. Разработка раздела «Добавление нового работника»…………………………….30

4.8. Разработка раздела «История входа»……………………………………………...32

4.9. Разработка раздела «Просмотреть работников»………………………………….35

4.10. Разработка раздела «Сформировать заказ»……………………………………...39

4.11. Разработка раздела «Просмотр клиентов»………………………………………44

4.12. Разработка раздела «Принять заказ»……………………………………………..48

5. Тестирование информационной системы…………………………………………………..52

6. Руководство пользователя…………………………………………………………………...54

6.1. Страница авторизации пользователя……………………………………………...54

6.2. Профиль сотрудника……………………………………………………………….54

6.3. Добавление новой услуги………………………………………………………….54

6.4. Регистрация нового клиента……………………………………………………….55

6.5. Добавление нового работника …………………………………………………….56

6.6. Просмотр истории входа…………………………………………………………...57

6.7. Просмотр информации о работниках……………………………………………..57

6.8. Формирование заказа………………………………………………………………57

6.9. Просмотр информации о клиентах………………………………………………..58

6.10. Принятие заказа…………………………………………………………………...59

7. Заключение…………………………………………………………………………………...60

Список использованных источников………………………………………………………….61

**Введение**

В настоящее время компьютерные технологии нашли широкое применение в различных областях. Огромное значение они имеют и в сфере сотовой связи. Работа современной системы сотовой связи не может быть возможна без использования компьютеров. Сейчас они используются уже не только как вычислительные машины, но и как средства связи.

Информационной системой (ИС) называется комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства, информационные ресурсы, а также персонал, обеспечивающий поддержку динамической информационной модели предметной области для удовлетворения информационных потребностей пользователей.

В ИС часть функций управления и обработки данных выполняется компьютерами, а часть человеком.

Современный мир информационных технологий трудно представить себе без возможности обработки больших объёмов информации. Такие объёмы информации удобно обрабатывать с помощью баз данных. Практически все системы в той или иной степени связаны с долговременным хранением и обработкой информации. Фактически, информация становится фактором, определяющим эффективность любой сферы деятельности. Увеличились информационные потоки и повысились требования к скорости обработки данных. Большинство операций не может быть выполнено вручную. Любые административные решения требуют более чёткой и точной оценки текущей ситуации и возможных перспектив её изменения.

В данной курсовой работе необходимо разработать модуль информационной системы для настольного клуба, который позволит выполнять управление и создавать заказы. Он должен обеспечивать просмотр, обработку и выборку данных из базы данных.

1. **Теоретические основы разрабатываемой темы**
   1. **Анализ проектируемой системы**

Вымышленный настольный клуб «6 Граней» является небольшой организацией, которая начала работать год назад. Организацией управляет администратор Федёров Иван Николаевич, в его подчинение входят 2 менеджера и два консультанта

«6 Граней» также располагает огромным выбором аренды на настольные игры. Цены на аренду выгодные.

Менеджер заходя в свой профиль способен сформировать новый заказ на аренду, добавить новую настольную игру в список или иной товар, и закрыть заказ на аренду, приняв настольные игры обратно.

Администратор после входа в профиль просмотреть информацию об истории входа сотрудников.

Консультант после входа в свой профиль может сформировать новый заказ на аренду, выбрав нужные товары, и закрыть заказ на аренду, приняв настольные игры обратно.

В данной курсовой работе будет разработан модуль автоматизированной информационной системы для настольного клуба «6 Граней». Она позволит сотрудникам клуба настольных игр производить различные операции с заказами на аренду, пользователями и другим функционалом программы.

* 1. **Обоснование выбора средств разработки информационной системы.**

Python – это универсальный современный ЯП высокого уровня, к преимуществам которого относят высокую производительность программных решений и структурированный, хорошо читаемый код. Синтаксис Питона максимально облегчен, что позволяет выучить его за сравнительно короткое время. Ядро имеет очень удобную структуру, а широкий перечень встроенных библиотек позволяет применять внушительный набор полезных функций и возможностей. ЯП может использоваться для написания прикладных приложений, а также разработки WEB-сервисов.

Python может поддерживать широкий перечень стилей разработки приложений, в том числе, очень удобен для работы с ООП и функционального программирования.

Один из самых популярных интерпретаторов языка – CPython, написанный на Си. Распространяется эта среда разработки бесплатно по свободной лицензии. Интерпретатор поддерживает большинство популярных платформ.

Питон активно развивается. Примерно раз в 2 года выходят обновления. Важной особенностью языка является отсутствие таких стандартов кодировки как ANSI, ISO и некоторых других, они работают благодаря интерпретатору.

ЯП имеет четко структурированное семантическое ядро и достаточно простой синтаксис. Все, что пишется на этом языке, всегда легко читаемо. В случае необходимости передать аргументы язык использует функцию call-by-sharing.

Набор операторов в языке вполне стандартен. Удобная особенность синтаксиса – это форматирование текста кода при помощи разбивки их на блоки с помощью отступов, которые создают нажатием клавиш «Space» и «Tab». В синтаксисе отсутствуют фигурные или операторные скобки, обозначающие начало и конец блока. Такое решение заметно сокращает количество строк тела программы и приучает программиста соблюдать хороший стиль и аккуратность при написании кода.

1. **Разработка технического задания**
   1. **Назначение и цели создания системы**

**2.1.1 Назначение системы**

Модуль информационной системы настольного клуба создается с целью обеспечения:

1. Создания менеджером и консультантом заказа и его формирование через добавление настольных игр в корзину с товарами, и присвоение этого заказа клиенту, вывод чека на печать, подсчёт стоимости за аренду;
2. Приём настольных игр из аренды обратно в настольный клуб.
3. Просмотра администратором истории входа сотрудников, добавление новых товаров.

**2.1.2 Цели создания системы**

1. Полный контроль над рабочей деятельности настольного клуба;
2. Учёт товаров, и информации о том, у каких клиентов они находятся в аренде;
3. Легкий способ получения полных отчетов информации о заказе на аренду.
   1. **Характеристика объектов автоматизации**

Автоматизируется процесс записи и хранения данных о товарах, сотрудниках и аренде в настольном клубе «6 Граней» и полной детализации его рабочей деятельности. Информация о сотрудниках, заказах и т.д. хранится в соответствующих таблицах БД и заносятся в них путем ввода данных через приложение, либо напрямую через базу данных.

* 1. **Требования к структуре системы**
     1. **Требования к структуре и функционированию системы**

В системе предлагается выделить следующие функциональные модули:

1. Модуль «Формирование заказов», который предназначен для хранения и обработки информации по конкретным заказам, либо датам заказов, а также для создания новых заказов.
2. Модуль «Выбор заказов», который предназначен для хранения и обработки информации по конкретным заказам и их статусам.
   * 1. **Требования к способам и средствам связи для информационного обмена между компонентами системы**

Взаимодействие компонентов системы осуществляется стандартными средствами платформы, на которой разработана система.

* + 1. **Требования к режимам функционирования системы**

Система должна поддерживать основной режим функционирования, в котором подсистемы приложения выполняют все свои основные функции.

В основном режиме функционирования Система должна обеспечивать:

1. работу пользователей режиме – 24 часов в день, 7 дней в неделю (24х7);
2. выполнение своих функций – сбор, обработка и загрузка данных; хранение данных, предоставление отчетности.
   * 1. **Требования по диагностированию системы**

Для обеспечения высокой надежности функционирования, как системы в целом, так и её отдельных компонентов должно обеспечиваться выполнение требований по диагностированию ее состояния.

* + 1. **Требования к надежности**

Уровень надежности должен достигаться согласованным применением организационных, организационно-технических мероприятий и программно-аппаратных средств.

При работе системы возможны следующие аварийные ситуации, которые влияют на надежность работы системы:

1. сбой в электроснабжении сервера;
2. сбой в электроснабжении рабочей станции пользователей системы;
3. сбой в электроснабжении обеспечения локальной сети (поломка сети);
4. сбои программного обеспечения сервера.

Проверка выполнения требований по надежности должна производиться на этапе проектирования расчетным путем, а на этапах испытаний и эксплуатации - по методике Разработчика, согласованной с Заказчиком.

* + 1. **Требования к защите информации**

1. Обеспечение информационное безопасности Системы должно удовлетворять следующим требованиям:
2. Защита Системы должна обеспечиваться комплексом программно-технических средств и поддерживающих их организационных мер.
3. Защита Системы должна обеспечиваться на всех технологических этапах обработки информации и во всех режимах функционирования, в том числе при проведении ремонтных и регламентных работ.
4. Программно-технические средства защиты не должны существенно ухудшать основные функциональные характеристики Системы (надежность, быстродействие, возможность изменения конфигурации).
   1. **Требования к функциям, выполняемым системой**
5. Хранение и обработка информации о сотрудниках;
6. Внесение изменений в БД;
   1. **Требования к видам обеспечения**
      1. **Требования к программному обеспечению**

Доступные аппаратные средства:

1. ПК с доступом в интернет.

Доступные программные средства:

1. PostgreSQL-сервер;
2. Web-браузер (Google Chrome, Mozilla Firefox и т.д.).
   * 1. **Требования к техническому обеспечению**

Система должна быть реализована с использованием специально выделенных серверов Заказчика.

1. **Проектирование информационной системы**

UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

* 1. **Диаграмма прецедентов (вариантов использования)**

Используется для представления функциональной модели информационной системы.

Прецеденты (варианты использования — Use Cases) — это подробные процедурные описания вариантов использования системы всеми заинтересованными лицами, а также внешними системами, т. е. всеми, кто (или что) может рассматриваться как актёры (actors) — действующие лица. По сути, это своего рода алгоритмы работы с системой с точки зрения внешнего мира.

Прецеденты являются основой функциональных требований к системе, позволяют описывать границы проектируемой системы, ее интерфейс, а затем выступают как основа для тестирования системы заказчиком с помощью приемочных тестов.

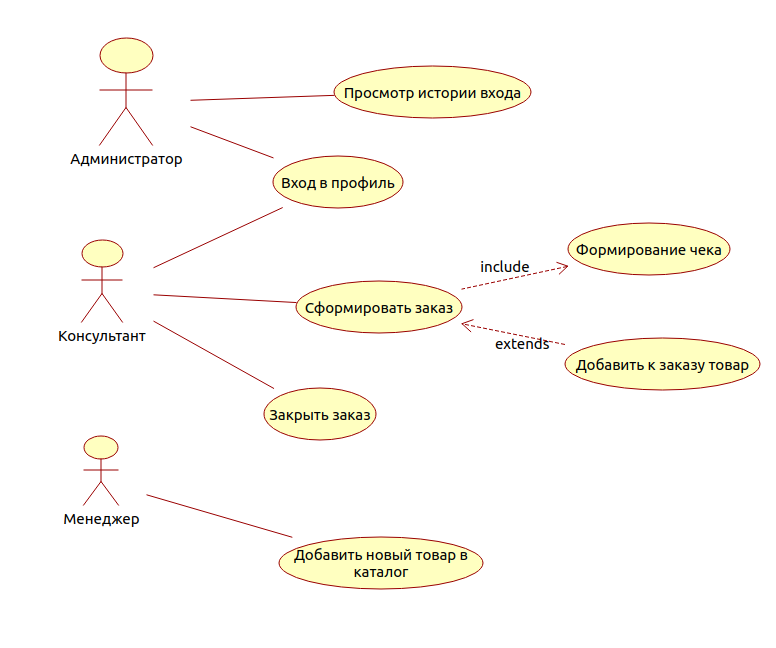


Рис. 1. Диаграмма прецедентов.

* 1. **Диаграмма последовательности**

Диаграмма последовательности — диаграмма, на которой показано взаимодействие объектов (обмен между ними сигналами и сообщениями), упорядоченное по времени, с отражением продолжительности обработки и последовательности их проявления.

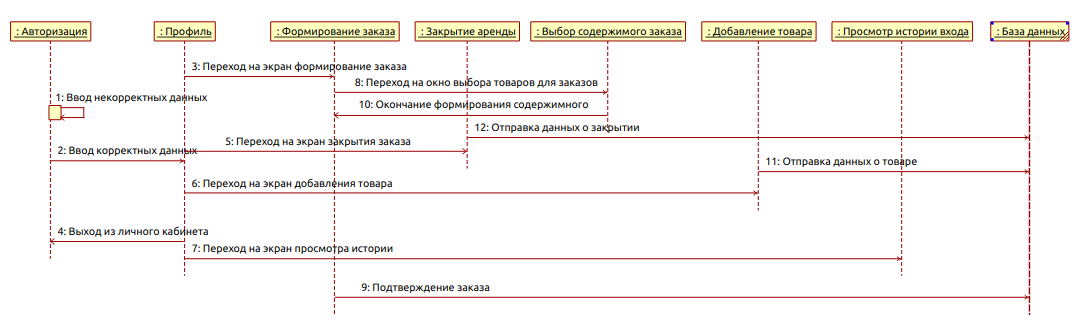


Рис. 2. Диаграмма последовательности

* 1. **Информационно логистическая модель базы данных**

Информационно-логическая модель отображает данные предметной области в виде совокупности информационных объектов и связей между ними. Эта модель представляет данные, подлежащие хранению в базе данных.

* + 1. **Описание структуры БД**

В нижеследующей модели описаны поля всех таблиц БД, используемых разработанной информационной системой.

Таблица 1. Описание атрибутов таблицы Клиенты

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  **п/п** | **Название** | **Идентификатор** | **Тип** | **Знач. по умолчанию** | **Обяз-ное поле?** | **Признак ключа** |
| 1. | Ид. № Клиента | id | N | – | + | PK |
| 2. | ФИО | fullname | С | – | + | - |
| 3. | Паспорт | passport | С | – | - | – |
| 4. | День рождения | birthday\_date | D | – | – | – |
| 5. | Адрес | address | C | - | - | – |
| 6. | Эл. почта | email | C | - | - | - |
| 7. | Пароль | password | C | - | - | - |

Таблица 2. Описание атрибутов таблицы Сотрудники

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  **п/п** | **Название** | **Идентификатор** | **Тип** | **Знач. по умолчанию** | **Обяз-ное поле?** | **Признак ключа** |
| 1. | Ид. № Сотрудника | id | N | – | + | PK |
| 2. | ФИО | fullname | С | – | + | - |
| 3. | Позиция | position | С | – | - | – |
| 4. | Последний вход | Last\_entry | D | – | – | – |
| 5. | Статус | status | C | - | - | – |
| 6. | Логин | login | C | - | - | - |
| 7. | Пароль | password | C | - | - | - |

Таблица 3. Описание атрибутов таблицы Товары

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  **п/п** | **Название** | **Идентификатор** | **Тип** | **Знач. по умолчанию** | **Обяз-ное поле?** | **Признак ключа** |
| 1. | Ид. № Товара | id | N | – | + | PK |
| 2. | Наименование | title | С | – | + | - |
| 3. | Изображение | img | С | – | - | – |
| 4. | Стоимость за час | cost\_per\_hour | N | – | – | – |
| 5. | Полная цена | full\_price | N | - | - | – |
| 6. | Описание | description | C | - | - | - |
| 7. | Оставшееся кол-во | remaining\_amount | N | - | - | - |
| 8. | Категория | category | C | - | - | - |

Таблица 4. Описание атрибутов таблицы Заказы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  **п/п** | **Название** | **Идентификатор** | **Тип** | **Знач. по умолчанию** | **Обяз-ное поле?** | **Признак ключа** |
| 1. | Ид. № Заказа | id | N | – | + | PK |
| 2. | Дата создания | create\_date | D | – | + | - |
| 3. | Время создания | create\_time | T | – | - | – |
| 4. | Ид. № Клиента | client\_id | N | – | – | FK(Клиенты) |
| 5. | Дата закрытия | close\_date | D | - | - | – |
| 6. | Время аренды | arenda\_hour\_time | N | - | - | - |
| 7. | Ид. № Сотрудника | employer\_id | N | - | - | FK(Сотрудники) |
| 8. | Статус | category | C | - | - | - |

Таблица 5. Описание атрибутов таблицы Заказ\_товары

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  **п/п** | **Название** | **Идентификатор** | **Тип** | **Знач. по умолчанию** | **Обяз-ное поле?** | **Признак ключа** |
| 1. | Ид. № Заказа | order\_id | N | – | + | PK, FK(Заказы) |
| 2. | Ид. № Товара | goods\_id | N | – | + | PK, FK(Товары) |

Таблица 6. Описание атрибутов таблицы История входа

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  **п/п** | **Название** | **Идентификатор** | **Тип** | **Знач. по умолчанию** | **Обяз-ное поле?** | **Признак ключа** |
| 1. | Ид. № Попытки входа | id | N | – | + | PK |
| 2. | Логин | login | С | – | + | – |
| 3. | Тип входа | entry\_type | C | - | - | - |
| 4. | Время входа | entry\_time | D | - | - | - |

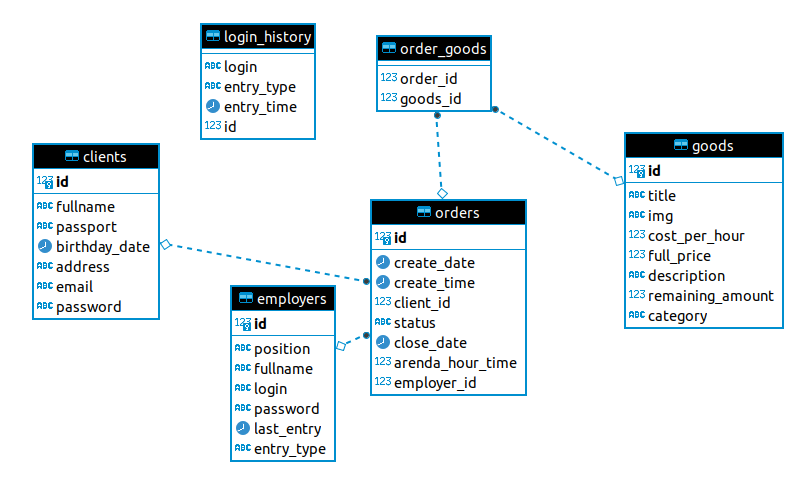


Рис. 3. Информационно-логистическая модель базы данных.

1. **Разработка информационной системы**
   1. **Описание структуры системы**

В магазине работают администратор, менеджер и консультант.

Консультант после входа в профиль может сформировать новый заказ, а также принять для закрытия старый.

Менеджер заходя в свой профиль может сформировать новый заказ, и принять для закрытия старый, и добавить новый товар.

Оператор заходя в свой профиль способен сформировать новый заказ, просмотреть информацию о клиентах и зарегистрировать нового клиента.

Администратор после входа в профиль может посмотреть информацию об истории входа сотрудников.

Система будет спроектирована с учётом паттерна MVC.

MVC расшифровывается как «модель-представление-контроллер» (от англ. model-view-controller). Это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения.

Компоненты MVC:

* Модель — этот компонент отвечает за данные, а также определяет структуру приложения. Например, если вы создаете To-Do приложение, код компонента model будет определять список задач и отдельные задачи.
* Представление — этот компонент отвечает за взаимодействие с пользователем. То есть код компонента view определяет внешний вид приложения и способы его использования.
* Контроллер — этот компонент отвечает за связь между model и view. Код компонента controller определяет, как сайт реагирует на действия пользователя. По сути, это мозг MVC-приложения.
  1. **Разработка файла подключения к базе данных**

Создаем класс DB.py. В этом файле опишем подключение к базе данных. И далее в процессе разработки использовать этот файл к другим python классам.

**Листинг класса DB.py**

class Db:

def \_\_init\_\_(self):

"""Инициализация соединения"""

self.connection = psycopg2.connect(

dbname='coursework\_prdb',

user='admin',

password='qwerty',

host='localhost'

)

self.connection.autocommit = True

self.cursor = self.connection.cursor(cursor\_factory=RealDictCursor)

self.simple\_cursor = self.connection.cursor()

print('successful connect to db')

db = Db()

* 1. **Разработка раздела «Авторизация пользователя»**

Данный раздел разработан с целью обеспечения разграничения доступа сотрудников к функционалу приложения. Суть заключается в сравнении данных, указанных пользователем на форме авторизации со значениями, хранящимися в базе и, в случае их совпадения, предоставление пользователю доступа к его личному кабинету. Создаем классы MVC для Authorize, в которых будет располагаться код основной формы авторизации. Так же создана функция скрытия/показа пароля, запрет ввода пароля, после трех неверных попыток ввода данных на 10 c. и вызов диалога с капчей

**Листинг класса AuthorizeController.py**

class AuthorizeController(QObject):

def \_\_init\_\_(self, view, model):

super().\_\_init\_\_()

self.\_model = model

self.\_view = view

@pyqtSlot(str, str)

def authorize(self, login, password):

"""авторизация"""

if not (login and password):

self.\_view.show\_error("Заполните все поля")

return

candidate = self.\_model.verify\_credentials(login, password)

if candidate:

model = PersonalAccountModel(candidate)

view = PersonalAccountView(model)

view.show()

self.\_view.close()

else:

self.\_view.show\_error("Неверный логин или пароль")

self.\_model.try\_auth += 1

@pyqtSlot()

def create\_capcha(self):

"""Создать капчу"""

self.\_view.on\_capcha()

from views.CapchaDialogView import CapchaDialogView

model = CapchaDialogModel()

view = CapchaDialogView(model, self.\_view)

view.show()

**Листинг класса AuthorizeModel.py**

class AuthorizeModel(QObject):

user\_authorized = pyqtSignal(bool)

block\_auth = pyqtSignal()

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.\_user\_model = UserModel()

self.\_cur\_user = None

self.\_try\_auth = 0

@property

def cur\_user(self):

return self.\_cur\_user

@cur\_user.setter

def cur\_user(self, value):

self.\_cur\_user = value

def verify\_credentials(self, login, password):

"""Проверить данные для входа"""

candidate = self.\_user\_model.get\_user(login, password)

input\_type = 'успешно' if candidate else 'не успешно'

time = datetime.datetime.now()

self.add\_to\_history(login, input\_type, time)

return candidate

def add\_to\_history(self, login, input\_type, time):

"""Добавить попытку в историю входа"""

db.simple\_cursor.execute(f'INSERT INTO login\_history '

f'("login", "entry\_type", "entry\_time") '

f'VALUES (%s, %s, %s)',

(

login,

input\_type,

time

))

# номер попытки авторизоваться

@property

def try\_auth(self):

return self.\_try\_auth

@try\_auth.setter

def try\_auth(self, value):

self.\_try\_auth = value

if self.\_try\_auth >= 4:

self.block\_auth.emit()

**Листинг класса AuthorizeView.py**

class AuthorizeView(QMainWindow):

def \_\_init\_\_(self, model):

super().\_\_init\_\_()

self.\_model = model

self.\_controller = AuthorizeController(self, self.\_model)

self.\_ui = Ui\_MainWindow()

self.setAttribute(Qt.WA\_DeleteOnClose)

self.\_ui.setupUi(self)

self.init\_slots()

self.init\_data()

def init\_slots(self):

"""Инициализация слотов"""

self.\_ui.show\_pass\_box.stateChanged.connect(self.on\_show\_pass\_changed)

self.\_ui.auth\_button.clicked.connect(self.on\_authorize\_click)

self.\_model.user\_authorized.connect(self.on\_authorize\_result)

self.\_model.block\_auth.connect(self.\_controller.create\_capcha)

def init\_data(self):

"""Инициализация данных"""

pass

# self.\_ui.login\_edit.setText("fedorov@namecomp.ru")

# self.\_ui.pass\_edit.setText("8ntwUp")

def show\_error(self, text):

"""Показать ошибку"""

msg\_box = QMessageBox()

msg\_box.setText(text)

msg\_box.setIcon(QMessageBox.Critical)

msg\_box.setWindowTitle("Ошибка авторизации")

msg\_box.exec()

@pyqtSlot(int)

def on\_show\_pass\_changed(self, is\_checked):

"""Слот для реакции на чекбокс о показе пароля"""

mode = QLineEdit.Normal if is\_checked else QLineEdit.Password

self.\_ui.pass\_edit.setEchoMode(mode)

@pyqtSlot()

def on\_authorize\_click(self):

"""Клик для авторизации"""

login = self.\_ui.login\_edit.text()

password = self.\_ui.pass\_edit.text()

self.\_controller.authorize(login, password)

@pyqtSlot(bool)

def on\_authorize\_result(self, value):

"""Результат авторизации"""

print(value)

def on\_capcha(self):

"""Слот при открытии капчи"""

self.\_ui.auth\_button.setDisabled(True)

def on\_block\_auth(self):

"""Слот при заблокировании возможности входа"""

timer = QTimer

self.\_ui.auth\_button.setDisabled(True)

timer.singleShot(10000, self.after\_block\_auth)

def on\_good\_capcha(self):

"""При успшном вводе капчи"""

self.\_ui.auth\_button.setDisabled(False)

def after\_block\_auth(self):

"""После блока возможности авторизоваться"""

self.\_ui.auth\_button.setDisabled(False)

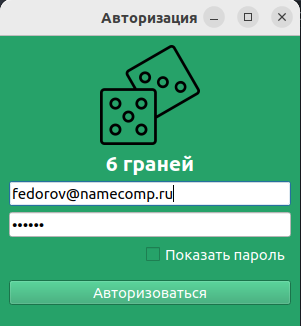


Рис. 4. Форма авторизации сотрудников

* 1. **Разработка раздела «Профиль сотрудника»**

Эту страницу будут видеть после успешной авторизации все сотрудники

службы такси. В зависимости от их должности определенные кнопки будут скрыты. Раздел профиль сотрудника является переходным для дальнейшего функционала в зависимости от должности сотрудника. Также в данном разделе сотруднику будет показано его ФИО, должность, а также его фотография.

**Листинг класса PersonalAccountController.py**

class PersonalAccountController(QObject):

def \_\_init\_\_(self, view, model):

super().\_\_init\_\_()

self.\_model = model

self.\_view = view

@pyqtSlot()

def open\_history(self):

"""Открыть окно с историей входа"""

model = LoginHistoryModel()

view = LoginHistoryView(model, self.\_view)

view.show()

@pyqtSlot()

def create\_order(self):

"""Открыть окно с созданием заказа"""

model = CreateOrderModel(self.\_model.user)

view = CreateOrderView(model, self.\_view)

view.show()

@pyqtSlot()

def close\_order(self):

"""Открыть окно с закрытием заказа"""

model = CloseOrderModel()

view = CloseOrderView(model, self.\_view)

view.show()

@pyqtSlot()

def add\_good(self):

"""Открыть окно с добавлением товара"""

model = AddGoodModel()

view = AddGoodView(model, self.\_view)

view.show()

@pyqtSlot()

def deauth(self):

"""Деавторизация"""

from views.AuthorizeView import AuthorizeView # for avoid circular import

model = AuthorizeModel()

view = AuthorizeView(model)

self.\_view.close()

view.show()

@pyqtSlot()

def close(self):

"""Закрытие экрана"""

self.\_view.close()

**Листинг класса PersonalAccountModel.py**

class PersonalAccountModel(QObject):

def \_\_init\_\_(self, user):

super().\_\_init\_\_()

self.\_user\_model = UserModel()

self.\_cur\_user = user

@property

def user(self):

"""Текущий пользователь"""

return self.\_cur\_user

**Листинг класса PersonalAccountView.py**

class PersonalAccountView(QMainWindow):

def \_\_init\_\_(self, model: PersonalAccountModel):

super().\_\_init\_\_()

self.\_model = model

self.\_controller = PersonalAccountController(self, self.\_model)

self.\_ui = Ui\_MainWindow()

self.setAttribute(Qt.WA\_DeleteOnClose)

self.\_ui.setupUi(self)

self.init\_slots()

self.init\_data()

def init\_slots(self):

"""Инициализация слотов"""

self.\_ui.see\_history.clicked.connect(self.\_controller.open\_history)

self.\_ui.create\_order\_btn.clicked.connect(self.\_controller.create\_order)

self.\_ui.deauth\_btn.triggered.connect(self.\_controller.deauth)

self.\_ui.exit\_btn.triggered.connect(self.\_controller.close)

self.\_ui.close\_order\_btn.clicked.connect(self.\_controller.close\_order)

self.\_ui.add\_good\_btn.clicked.connect(self.\_controller.add\_good)

# self.\_ui.create\_order\_btn.clicked.connect(self.\_controller)

def init\_data(self):

"""Инициализация данных"""

fio = self.\_model.user['fullname']

position = self.\_model.user['position']

lastname = fio.split()[0]

photo = QPixmap(f"./img/{lastname}.jpeg")

self.\_ui.photo\_label.setPixmap(photo)

self.\_ui.fio\_label.setText(fio)

self.\_ui.position\_label.setText(position)

if position != 'Администратор':

self.\_ui.see\_history.hide()

else:

self.\_ui.create\_order\_btn.hide()

self.\_ui.close\_order\_btn.hide()

if position != 'Менеджер':

self.\_ui.add\_good\_btn.hide()

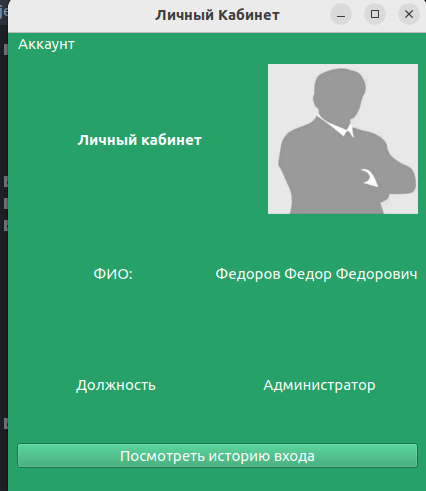


Рис. 5. Окно профиля менеджера

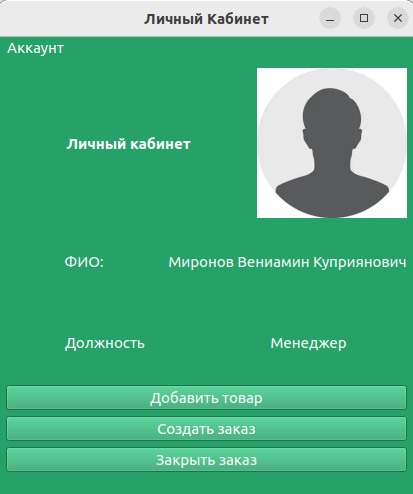


Рис. 6. Окно профиля менеджера

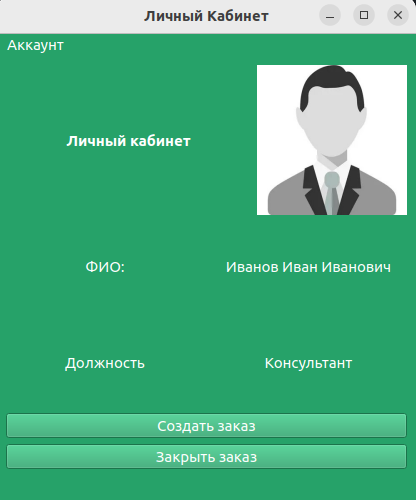


Рис. 7. Окно профиля консультанта

* 1. **Разработка раздела «Добавление нового товара»**

Данный раздел нужен для добавления нового товара в базу данных настольного клуба. Этим функционалом может пользоваться только менеджер.

**Листинг класса AddGoodController.py**

class AddGoodController(QObject):

def \_\_init\_\_(self, view, model):

super().\_\_init\_\_()

self.\_model = model

self.\_view = view

@pyqtSlot(dict)

def add\_good(self, good):

"""Добавить товар"""

self.\_model.add\_good(good)

@pyqtSlot()

def on\_added\_good(self):

"""Слот при успшеном добавлении товара"""

self.\_view.close()

**Листинг класса AddGoodModel.py**

class AddGoodModel(QObject):

good\_added = pyqtSignal()

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.db = db

def get\_last\_good\_id(self):

self.db.cursor.execute('SELECT MAX("id") FROM goods')

result = self.db.cursor.fetchone()['max']

return result

def add\_good(self, good):

if '' in (good['title'], good['hour\_cost'], good['full\_cost'], good['category']):

return

good['id'] = self.get\_last\_good\_id() + 1

db.cursor.execute(f'insert into goods '

f'VALUES '

f'('

f'%s, %s, %s, %s, %s, %s, %s, %s'

f')',

(

good['id'],

good['title'],

good['img'],

good['hour\_cost'],

good['full\_cost'],

good['description'],

good['count'],

good['category']

)

)

self.good\_added.emit()

**Листинг класса AddGoodView.py**

class AddGoodModel(QObject):

good\_added = pyqtSignal()

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.db = db

def get\_last\_good\_id(self):

"""Получение id последнего товара"""

self.db.cursor.execute('SELECT MAX("id") FROM goods')

result = self.db.cursor.fetchone()['max']

return result

def add\_good(self, good):

"""Добавить товар в БД1"""

if '' in (good['title'], good['hour\_cost'], good['full\_cost'], good['category']):

return

good['id'] = self.get\_last\_good\_id() + 1

db.cursor.execute(f'insert into goods '

f'VALUES '

f'('

f'%s, %s, %s, %s, %s, %s, %s, %s'

f')',

(

good['id'],

good['title'],

good['img'],

good['hour\_cost'],

good['full\_cost'],

good['description'],

good['count'],

good['category']

)

)

self.good\_added.emit()

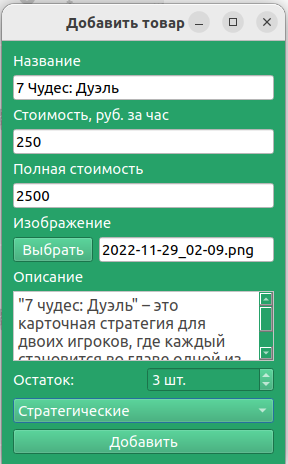


Рис. 8. Окно раздела «Добавление товара»

* 1. **Разработка раздела «Регистрация нового клиента»**

Данным функционалом может обладать Администратор и оператор. Текущий раздел нужен для добавления новой услуги в базу данных службы такси.

**Листинг класса Registration.java**

public class Registration implements Initializable {

@FXML

private TextField clientId;

@FXML

private TextField clientFio;

@FXML

private TextField clientAddress;

@FXML

private TextField clientBirth;

@FXML

private TextField clientLogin;

@FXML

private TextField clientPassword;

@FXML

private Button registerBtn;

@FXML

private Label regLabel;

private String id = "";

private String fio = "";

private String address = "";

private String birth = "";

private String login = "";

private String password = "";

private final String QUERY= "insert into clients(client\_id, fio, address, birthday, client\_login, client\_password) values(?, ?, ?, ?, ?, ?)";

@Override

public void initialize(URL location, ResourceBundle resources) {

}

//Метод для регистрации нового клиента

public void clientRegistration() throws SQLException {

id=clientId.getText();

fio=clientFio.getText();

address=clientAddress.getText();

birth=clientBirth.getText();

login=clientLogin.getText();

password=clientPassword.getText();

Connection connection = DriverManager.getConnection(DATABASE\_URL, DATABASE\_USERNAME, DATABASE\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement(QUERY);

preparedStatement.setString(1, id);

preparedStatement.setString(2, fio);

preparedStatement.setString(3, address);

preparedStatement.setString(4, birth);

preparedStatement.setString(5, login);

preparedStatement.setString(6, password);

try {

preparedStatement.executeUpdate();

} catch (SQLException e) {

regLabel.setTextFill(Color.web("#a63f3f"));

regLabel.setText("Ошибка регистрации");

throw new RuntimeException(e);

}

regLabel.setTextFill(Color.web("#3fa655"));

regLabel.setText("Клиент зарегистрирован");

}

}

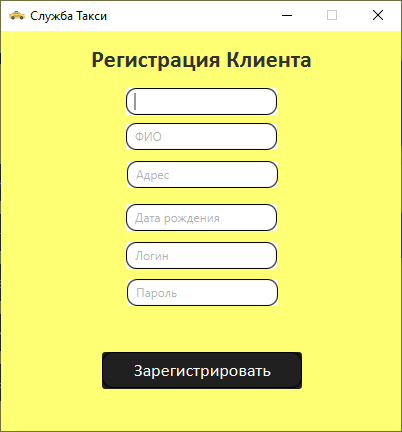


Рис. 9. Окно раздела «Регистрация клиента»

* 1. **Разработка раздела «Добавление нового работника»**

Данный раздел предназначен только для администратора. В нем он может добавить нового работника в базу данных службы такси.

**Листинг класса addEmployee.java**

public class AddEmployee implements Initializable {

@FXML

private TextField EmployeeId;

@FXML

private TextField EmployeeFio;

@FXML

private TextField EmployeePost;

@FXML

private TextField EmployeeAddress;

@FXML

private TextField EmployeeBirth;

@FXML

private TextField EmployeeLogin;

@FXML

private TextField EmployeePassword;

@FXML

private Button EmployeeBtn;

@FXML

private Label employeeLabel;

private String id = "";

private String fio = "";

private String post = "";

private String address = "";

private String birth = "";

private String login = "";

private String password = "";

private final String QUERY= "insert into employee(employee\_id, FIO, post, address, birthday, login, password) values(?, ?, ?, ?, ?, ?, ?)";

@Override

public void initialize(URL location, ResourceBundle resources) {

}

//Метод для добавления нового работника

public void addEmployee() throws SQLException {

id=EmployeeId.getText();

fio=EmployeeFio.getText();

post = EmployeePost.getText();

address=EmployeeAddress.getText();

birth=EmployeeBirth.getText();

login=EmployeeLogin.getText();

password=EmployeePassword.getText();

Connection connection = DriverManager.getConnection(DATABASE\_URL, DATABASE\_USERNAME, DATABASE\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement(QUERY);

preparedStatement.setString(1, id);

preparedStatement.setString(2, fio);

preparedStatement.setString(3,post);

preparedStatement.setString(4, address);

preparedStatement.setString(5, birth);

preparedStatement.setString(6, login);

preparedStatement.setString(7, password);

try {

preparedStatement.executeUpdate();

} catch (SQLException e) {

employeeLabel.setTextFill(Color.web("#a63f3f"));

employeeLabel.setText("Ошибка регистрации");

throw new RuntimeException(e);

}

employeeLabel.setTextFill(Color.web("#3fa655"));

employeeLabel.setText("Работник зарегистрирован");

}

}

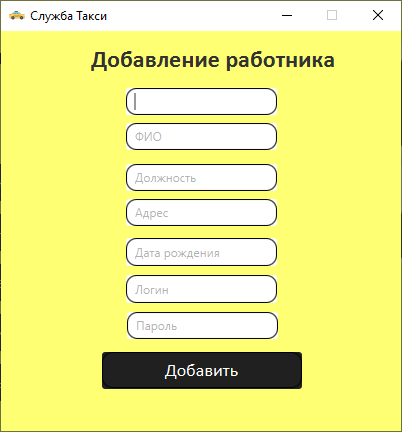


Рис. 10. Окно раздела «Добавление работника»

* 1. **Разработка раздела «История входа»**

Данный раздел предназачен только для администратора. В текущем разделе реализован функционал для просмотра работников по истории входа и последующего формирования и скачивания отчета.

**Листинг класса History.java**

public class History implements Initializable {

private ObservableList<HistData> history = FXCollections.observableArrayList();

@FXML

private Button saveBtn;

@FXML

private Label statusOtchet;

@FXML

private TableView<HistData> tableView;

@FXML

private TableColumn<HistData, String> tableColumn1;

@FXML

private TableColumn<HistData, String> tableColumn2;

//for pdf

private String data;

private String fio;

private String lastEnter;

//Метод инициализации формы с таблицей истории входа

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

//Вывод записей в таблицу

ArrayList<String> name = null;

try {

name = DB.getHist();

} catch (SQLException e) {

throw new RuntimeException(e);

}

for(int i = 0; i < name.size(); ++i){

history.add(new HistData(name.get(i), name.get(++i)));

}

//Задаем тип данных переменным

tableColumn1.setCellValueFactory(new PropertyValueFactory<HistData, String>("fio"));

tableColumn2.setCellValueFactory(new PropertyValueFactory<HistData, String>("enter"));

tableView.setItems(history);

}

//Метод для показа кнопки saveBtn

public void formOtchet(ActionEvent actionEvent) throws SQLException, ClassNotFoundException {

saveBtn.setVisible(true);

}

//Метод для сохранения отчета

public void saveOtchet(ActionEvent actionEvent) throws FileNotFoundException, DocumentException, SQLException {

Connection connection = DriverManager.getConnection(DATABASE\_URL, DATABASE\_USERNAME, DATABASE\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement("SELECT FIO, last\_enter FROM `employee`;");

ResultSet resultSet = preparedStatement.executeQuery();

while(resultSet.next()) {

fio = resultSet.getString(1);

lastEnter = resultSet.getString(2);

}

connection.close();

Document document = new Document();

PdfWriter.getInstance(document, new FileOutputStream("fileHistory.pdf"));

Font f1 = FontFactory.getFont("DejaVuSans.ttf", "cp1251", BaseFont.EMBEDDED, 10);

document.open();

ArrayList<String> name = null;

try {

name = DB.getHist();

} catch (SQLException e) {

throw new RuntimeException(e);

}

for(int i = 0; i < name.size(); i++){

List list = new List();

list.add((new ListItem("FIO: "+ name.get(i) +"\t", f1)));

list.add((new ListItem("Last Enter: "+name.get(++i)+"\t\n\n", f1)));

document.add(list);

}

document.close();

statusOtchet.setVisible(true);

}

}

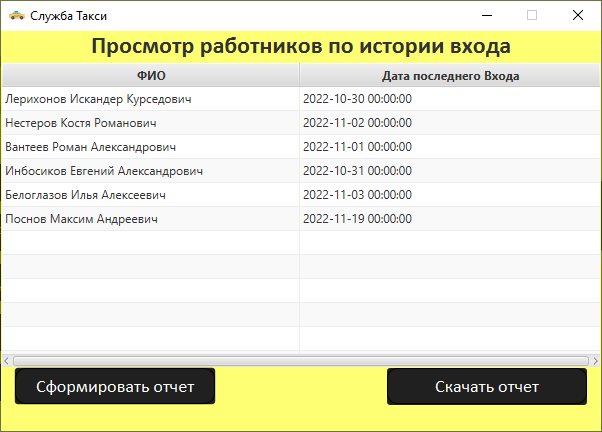


Рис. 11. Окно раздела «Просмотр работников по истории входа»

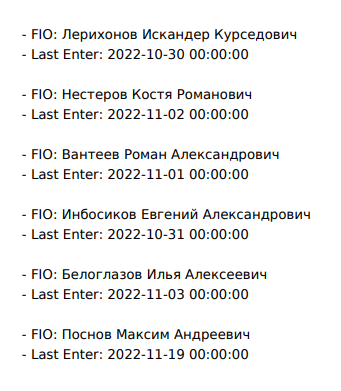


Рис. 12. Отчет по «Просмотр работников по истории входа»

* 1. **Разработка раздела «Посмотреть работников»**

В данном разделе только администратор может посмотреть всю информацию о работниках предприятия службы такси «UberNN» включая их логины, пароли и личную информацию, и в последующем сформировать и скачать отчет.

**Листинг класса showEmployee.java**

public class showEmployee implements Initializable {

private ObservableList<empData> emp = FXCollections.observableArrayList();

@FXML

private Button saveBtn;

@FXML

private Button histExtBtn;

@FXML

private Label statusOtchet;

@FXML

private TableView<empData> tableViewEmp;

@FXML

private TableColumn<empData, String> tableColumnId;

@FXML

private TableColumn<empData, String> tableColumnFIO;

@FXML

private TableColumn<empData, String> tableColumnPost;

@FXML

private TableColumn<empData, String> tableColumnAddress;

@FXML

private TableColumn<empData, String> tableColumnBirthday;

@FXML

private TableColumn<empData, String> tableColumnLogin;

@FXML

private TableColumn<empData, String> tableColumnPassword;

@FXML

private TableColumn<empData, String> tableColumnEnter;

//for pdf

private String data;

private String id;

private String fio;

private String post;

private String address;

private String birthday;

private String login;

private String password;

private String lastEnter;

//Метод инициализации формы с таблицей истории входа

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

//Вывод записей в таблицу

ArrayList<String> name = null;

try {

name = DB.getEmp();

} catch (SQLException e) {

throw new RuntimeException(e);

}

for(int i = 0; i < name.size(); ++i){

emp.add(new empData(name.get(i), name.get(++i), name.get(++i), name.get(++i), name.get(++i), name.get(++i), name.get(++i), name.get(++i)));

}

//Задаем тип данных переменным

tableColumnId.setCellValueFactory(new PropertyValueFactory<empData, String>("id"));

tableColumnFIO.setCellValueFactory(new PropertyValueFactory<empData, String>("fio"));

tableColumnPost.setCellValueFactory(new PropertyValueFactory<empData, String>("post"));

tableColumnAddress.setCellValueFactory(new PropertyValueFactory<empData, String>("address"));

tableColumnBirthday.setCellValueFactory(new PropertyValueFactory<empData, String>("birthday"));

tableColumnLogin.setCellValueFactory(new PropertyValueFactory<empData, String>("login"));

tableColumnPassword.setCellValueFactory(new PropertyValueFactory<empData, String>("password"));

tableColumnEnter.setCellValueFactory(new PropertyValueFactory<empData, String>("enter"));

tableViewEmp.setItems(emp);

}

//Метод для показа кнопки saveBtn

public void formOtchet(ActionEvent actionEvent) throws SQLException, ClassNotFoundException {

saveBtn.setVisible(true);

}

//Метод для сохранения отчета

public void saveOtchet(ActionEvent actionEvent) throws FileNotFoundException, DocumentException, SQLException {

Connection connection = DriverManager.getConnection(DATABASE\_URL, DATABASE\_USERNAME, DATABASE\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement("SELECT \* FROM `employee`;");

ResultSet resultSet = preparedStatement.executeQuery();

while(resultSet.next()) {

id = resultSet.getString(1);

fio = resultSet.getString(2);

post = resultSet.getString(3);

address = resultSet.getString(4);

birthday = resultSet.getString(5);

login = resultSet.getString(6);

password = resultSet.getString(7);

lastEnter = resultSet.getString(8);

}

connection.close();

Document document = new Document();

PdfWriter.getInstance(document, new FileOutputStream("fileEmployee.pdf"));

Font f1 = FontFactory.getFont("DejaVuSans.ttf", "cp1251", BaseFont.EMBEDDED, 10);

document.open();

ArrayList<String> name = null;

try {

name = DB.getEmp();

} catch (SQLException e) {

throw new RuntimeException(e);

}

for(int i = 0; i < name.size(); i++){

List list = new List();

list.add((new ListItem("ID: "+ name.get(i) +"\t", f1)));

list.add((new ListItem("FIO: "+ name.get(++i) +"\t", f1)));

list.add((new ListItem("Post: "+ name.get(++i) +"\t", f1)));

list.add((new ListItem("Address: "+ name.get(++i) +"\t", f1)));

list.add((new ListItem("Birthday: "+ name.get(++i) +"\t", f1)));

list.add((new ListItem("Login: "+ name.get(++i) +"\t", f1)));

list.add((new ListItem("Password: "+ name.get(++i) +"\t", f1)));

list.add((new ListItem("Last Enter: "+name.get(++i)+"\t\n\n", f1)));

document.add(list);

}

System.out.println(name);

document.close();

statusOtchet.setVisible(true);

}

}

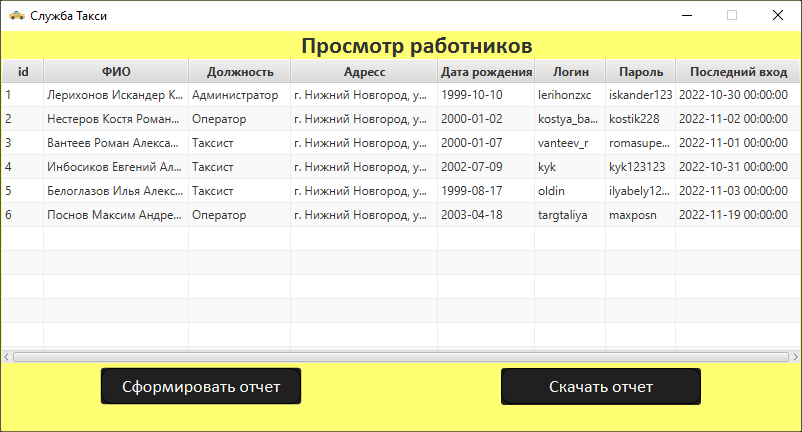


Рис. 13. Окно раздела «Просмотр работников»



Рис. 14. Отчет по разделу «Просмотр работников»

* 1. **Разработка раздела «Сформировать заказ»**

Данный раздел предназначен для оператора. В нем он может создавать новый заказ, данные которого будут занесены в базу данных службы такси.

Пустыми значениями в базе данных у данного заказа останутся только статус заказа и время заказа.

Также в данном разделе оператор может сформировать и скачать штрих-код к заказу.

**Листинг класса formOrder.java**

public class formOrder implements Initializable {

@FXML

private TextField orderId;

@FXML

private TextField orderDate;

@FXML

private TextField orderTimeDate;

@FXML

private TextField orderClientId;

@FXML

private TextField orderServiceId;

@FXML

private Label orderLabel;

@FXML

private Canvas canvas;

@FXML

private Label savePdf;

private String id = "";

private String date = "";

private String timeDate = "";

private String clientId = "";

private String serviceId = "";

private Window primaryStage;

private final String QUERY= "insert into orders(order\_id, date\_order, time\_date, client\_id, service\_id) values(?, ?, ?, ?, ?)";

private static final int CANVAS\_SIZE = 200;

@Override

public void initialize(URL location, ResourceBundle resources) {

}

//Метод для формирования заказа

public void formOrder() throws SQLException {

id=orderId.getText();

date=orderDate.getText();

timeDate=orderTimeDate.getText();

clientId=orderClientId.getText();

serviceId=orderServiceId.getText();

Connection connection = DriverManager.getConnection(DATABASE\_URL, DATABASE\_USERNAME, DATABASE\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement(QUERY);

preparedStatement.setString(1, id);

preparedStatement.setString(2, date);

preparedStatement.setString(3, timeDate);

preparedStatement.setString(4, clientId);

preparedStatement.setString(5, serviceId);

try {

preparedStatement.executeUpdate();

} catch (SQLException e) {

orderLabel.setTextFill(Color.web("#a63f3f"));

orderLabel.setText("Ошибка формирования");

throw new RuntimeException(e);

}

orderLabel.setTextFill(Color.web("#3fa655"));

orderLabel.setText("Заказ сформирован");

}

//Метод для формирования штрих-кода

public void loadCode(ActionEvent actionEvent) {

if(orderDate.getText().equals("") || orderTimeDate.getText().equals("")){

System.out.println("Введены не все данные");

}else {

GraphicsContext gc = canvas.getGraphicsContext2D();

gc.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());

double mm = (double) Toolkit.getDefaultToolkit().getScreenResolution() / 25.4D;

System.out.println(mm);

int[] palks = new int[]{5, 1, 4, 0, 9, 2, 0, 2, 0, 1, 2, 3, 4, 5, 6};

gc.setFill(Color.BLACK);

int x0 = 20;

int y0 = 10;

double heightPalks = 22.85D \* mm;

double weightOgrPalks = 0.5D \* mm;

double rasstoyanieMegdyPalk = 0.5D \* mm;

gc.fillRect((double)x0, (double)y0, weightOgrPalks, heightPalks + 1.65D \* mm);

int otcydaPalks = (int)((double)x0 + weightOgrPalks + rasstoyanieMegdyPalk);

int otcydaZifra = otcydaPalks;

gc.fillRect((double)otcydaPalks, (double)y0, weightOgrPalks, heightPalks + 1.65D \* mm);

otcydaPalks = (int)((double)otcydaPalks + weightOgrPalks + rasstoyanieMegdyPalk);

boolean printSrednyaPalks = false;

for(int numberPalks = 0; numberPalks < palks.length; ++numberPalks) {

double shirinaPalks = (double)palks[numberPalks] \* 0.15D \* mm;

if (numberPalks == palks.length / 2 && !printSrednyaPalks) {

gc.setFill(Color.BLACK);

gc.fillRect((double)otcydaPalks, (double)y0, weightOgrPalks, heightPalks + 1.65D \* mm);

otcydaPalks = (int)((double)otcydaPalks + weightOgrPalks + rasstoyanieMegdyPalk);

gc.fillRect((double)otcydaPalks, (double)y0, weightOgrPalks, heightPalks + 1.65D \* mm);

otcydaPalks = (int)((double)otcydaPalks + weightOgrPalks + rasstoyanieMegdyPalk);

--numberPalks;

printSrednyaPalks = true;

} else {

if (shirinaPalks == 0.0D) {

shirinaPalks = 1.35D \* mm;

gc.setFill(Color.WHITE);

} else {

gc.setFill(Color.BLACK);

}

gc.fillRect((double)otcydaPalks, (double)y0, shirinaPalks, heightPalks);

otcydaPalks = (int)((double)otcydaPalks + shirinaPalks + rasstoyanieMegdyPalk);

}

}

gc.fillRect((double)otcydaPalks, (double)y0, weightOgrPalks, heightPalks + 1.65D \* mm);

otcydaPalks = (int)((double)otcydaPalks + weightOgrPalks + rasstoyanieMegdyPalk);

gc.fillRect((double)otcydaPalks, (double)y0, weightOgrPalks, heightPalks + 1.65D \* mm);

String timeStamp = new SimpleDateFormat("ddMMyyyyHHmm").format(Calendar.getInstance().getTime());

String codes = orderDate.getText()+timeStamp+orderTimeDate.getText();

for(int i = 0; i < 6; i++){

int a = (int) (Math.random()\*10);

codes+= a;

}

gc.fillText(codes, x0, y0+heightPalks + 1.65D \* mm + 10, 100);

}

}

//Метод для скачивания штрих-кода в pdf формат

public void pdf() throws URISyntaxException, IOException, DocumentException {

FileChooser savefile = new FileChooser();

savefile.setTitle("Save File");

File file = new File("src/main/resources/shtrih-code.jpeg");

if (file != null) {

try {

WritableImage writableImage = new WritableImage(CANVAS\_SIZE, CANVAS\_SIZE);

canvas.snapshot(null, writableImage);

RenderedImage renderedImage = SwingFXUtils.fromFXImage(writableImage, null);

ImageIO.write(renderedImage, "png", file);

} catch (IOException ex) {

ex.printStackTrace();

System.out.println("Error!");

}

}

Path path1 = Paths.get(ClassLoader.getSystemResource("shtrih-code.jpeg").toURI());

File file1 = savefile.showSaveDialog(primaryStage);

Document document1 = new Document();

PdfWriter.getInstance(document1, new FileOutputStream(String.valueOf(file1)));

document1.open();

com.itextpdf.text.Image image = com.itextpdf.text.Image.getInstance(path1.toAbsolutePath().toString());

document1.add(image);

document1.close();

savePdf.setVisible(true);

}

}

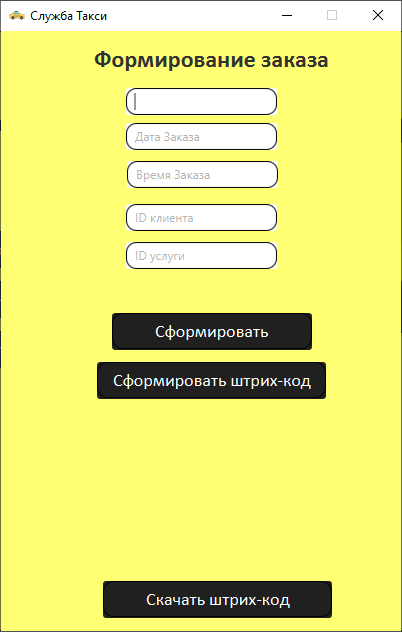


Рис. 15. Окно раздела «Формирование заказа»



Рис. 16. Пример полученного штрих-кода

* 1. **Разработка раздела «Просмотр клиентов»**

Данный раздел предназначен только для оператора, так как именно он контактируем с клиентами чаще всего. В нем он может посмотреть данные клиентов, сформировать отчет и в последующем его скачать. Данные о клиентах берутся с базы данных предприятия службы такси «UberNN».

**Листинг класса showClients.java**

public class showClients implements Initializable {

private ObservableList<clientData> clients = FXCollections.observableArrayList();

@FXML

private Button saveBtn;

@FXML

private Label statusOtchet;

@FXML

private TableView<clientData> tableViewClients;

@FXML

private TableColumn<clientData, String> tableColumnId;

@FXML

private TableColumn<clientData, String> tableColumnFIO;

@FXML

private TableColumn<clientData, String> tableColumnAddress;

@FXML

private TableColumn<clientData, String> tableColumnBirthday;

@FXML

private TableColumn<clientData, String> tableColumnLogin;

@FXML

private TableColumn<clientData, String> tableColumnPassword;

//for pdf

private String id;

private String fio;

private String address;

private String birthday;

private String login;

private String password;

//Метод инициализации формы с таблицей истории входа

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

//Вывод записей в таблицу

ArrayList<String> name = null;

try {

name = DB.getClients();

} catch (SQLException e) {

throw new RuntimeException(e);

}

for (int i = 0; i < name.size(); ++i) {

clients.add(new clientData(name.get(i), name.get(++i), name.get(++i), name.get(++i), name.get(++i), name.get(++i)));

}

//Задаем тип данных переменным

tableColumnId.setCellValueFactory(new PropertyValueFactory<clientData, String>("id"));

tableColumnFIO.setCellValueFactory(new PropertyValueFactory<clientData, String>("fio"));

tableColumnAddress.setCellValueFactory(new PropertyValueFactory<clientData, String>("address"));

tableColumnBirthday.setCellValueFactory(new PropertyValueFactory<clientData, String>("birthday"));

tableColumnLogin.setCellValueFactory(new PropertyValueFactory<clientData, String>("login"));

tableColumnPassword.setCellValueFactory(new PropertyValueFactory<clientData, String>("password"));

tableViewClients.setItems(clients);

}

//Метод для показа кнопки saveBtn

public void formOtchet(ActionEvent actionEvent) throws SQLException, ClassNotFoundException {

saveBtn.setVisible(true);

}

//Метод для сохранения отчета

public void saveOtchet(ActionEvent actionEvent) throws FileNotFoundException, DocumentException, SQLException {

Connection connection = DriverManager.getConnection(DATABASE\_URL, DATABASE\_USERNAME, DATABASE\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement("SELECT \* FROM `clients`;");

ResultSet resultSet = preparedStatement.executeQuery();

while(resultSet.next()) {

id = resultSet.getString(1);

fio = resultSet.getString(2);

address = resultSet.getString(3);

birthday = resultSet.getString(4);

login = resultSet.getString(5);

password = resultSet.getString(6);

}

connection.close();

Document document = new Document();

PdfWriter.getInstance(document, new FileOutputStream("fileClients.pdf"));

Font f1 = FontFactory.getFont("DejaVuSans.ttf", "cp1251", BaseFont.EMBEDDED, 10);

document.open();

ArrayList<String> name = null;

try {

name = DB.getClients();

} catch (SQLException e) {

throw new RuntimeException(e);

}

for(int i = 0; i < name.size(); i++){

List list = new List();

list.add((new ListItem("ID: "+ name.get(i) +"\t", f1)));

list.add((new ListItem("FIO: "+ name.get(++i) +"\t", f1)));

list.add((new ListItem("Address: "+ name.get(++i) +"\t", f1)));

list.add((new ListItem("Birthday: "+ name.get(++i) +"\t", f1)));

list.add((new ListItem("Login: "+ name.get(++i) +"\t", f1)));

list.add((new ListItem("Password: "+ name.get(++i) +"\t\n\n", f1)));

document.add(list);

}

System.out.println(name);

document.close();

statusOtchet.setVisible(true);

}

}

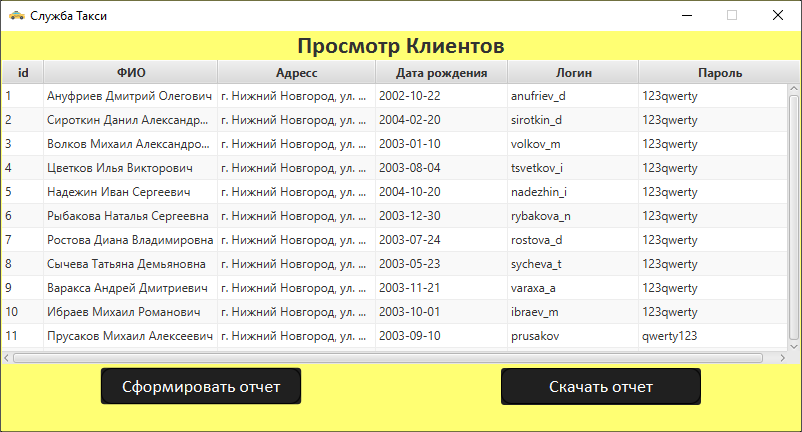


Рис. 17. Окно раздела «Просмотр клиентов»

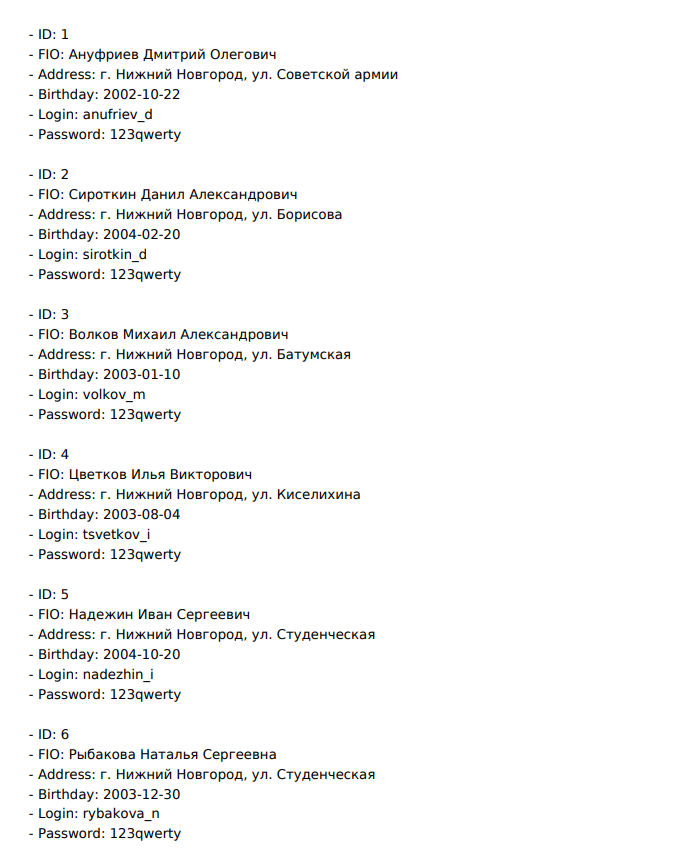


Рис. 18. Отчет по разделу «Просмотр клиентов»

* 1. **Разработка раздела «Принять заказ»**

Данный раздел предназначен только для должности «таксист». В нем таксисты могут выбирать непринятые заказы для последующего их завершения или отмены.

**Листинг класса enterOrder.java**

public class enterOrder implements Initializable {

@FXML

private ComboBox orderSelect;

@FXML

private TextField orderTime;

@FXML

private Button confirmBtn;

@FXML

private Button cancelBtn;

@FXML

private Label orderLabel;

private String buffer = "";

String[]arr = new String[3];

private String Order = "";

private String time = "";

private String state = "";

private final String QUERY= "insert into orders(state\_order, time\_order) values(?, ?)";

private final String QUERY2 = "SELECT order\_id FROM `orders` WHERE orders.state\_order is NULL and orders.time\_order is NULL;";

//Метод для выбора невыполненных заказов

@Override

public void initialize(URL url, ResourceBundle resourceBundle){

Connection connection = null;

PreparedStatement preparedStatement = null;

ResultSet resultSet = null;

try {

connection = DriverManager.getConnection(DATABASE\_URL, DATABASE\_USERNAME, DATABASE\_PASSWORD);

preparedStatement = connection.prepareStatement(QUERY2);

resultSet = preparedStatement.executeQuery();

while (resultSet.next()){

orderSelect.getItems().add(resultSet.getString(1));

}

} catch (SQLException e) {

throw new RuntimeException(e);

}

}

//Метод для завершения заказа

public void confirmOrder() throws SQLException {

buffer = orderSelect.getValue().toString();

arr = buffer.split(",");

Order = arr[0];

System.out.println(Order);

time=orderTime.getText();

state = "Завершено";

Connection connection = DriverManager.getConnection(DATABASE\_URL, DATABASE\_USERNAME, DATABASE\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement("update orders set " +

"state\_order =\""+state+"\", time\_order ="+time+" where order\_id=" +Order+"");

try {

preparedStatement.executeUpdate();

} catch (SQLException e) {

orderLabel.setTextFill(Color.web("#a63f3f"));

orderLabel.setText("Ошибка");

throw new RuntimeException(e);

}

orderLabel.setTextFill(Color.web("#3fa655"));

orderLabel.setText("Завершено");

}

//Метод для отмены заказа

public void cancelOrder() throws SQLException {

buffer = orderSelect.getValue().toString();

arr = buffer.split(",");

Order = arr[0];

System.out.println(Order);

time=orderTime.getText();

state = "Отменено";

Connection connection = DriverManager.getConnection(DATABASE\_URL, DATABASE\_USERNAME, DATABASE\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement("update orders set " +

"state\_order =\""+state+"\", time\_order ="+time+" where order\_id=" +Order+"");

try {

preparedStatement.executeUpdate();

} catch (SQLException e) {

orderLabel.setTextFill(Color.web("#a63f3f"));

orderLabel.setText("Ошибка");

throw new RuntimeException(e);

}

orderLabel.setTextFill(Color.web("#3fa655"));

orderLabel.setText("Заказ отменен");

}

}

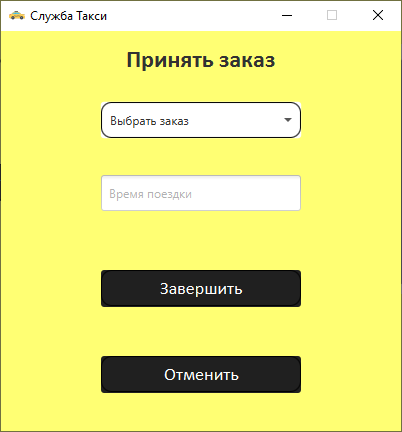


Рис. 19. Окно раздела «Принять заказ»

1. **Тестирование информационной системы**

Тестирование – важная часть любой программы контроля качества, поэтому информационная система пройдет этап тестирования, чтобы выявить и искоренить ошибки, мешающие комфортной работе с системой. В момент тестирования приложения были обнаружены некоторые проблемы и неточности.

При создании отчета остатков товаров, в таблице, куда заносились данные, не распознавались русские символы.

Причина: В дополнительной библиотеке, которая использовалась для переноса данных в pdf файл, могут использоваться только английские символы.

Код с ошибкой:

public void saveOtchet(ActionEvent actionEvent) throws FileNotFoundException, DocumentException, SQLException {

Connection connection = DriverManager.getConnection(DATABASE\_URL, DATABASE\_USERNAME, DATABASE\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement("SELECT FIO, last\_enter FROM `employee`;");

ResultSet resultSet = preparedStatement.executeQuery();

while(resultSet.next()) {

fio = resultSet.getString(1);

lastEnter = resultSet.getString(2);

}

connection.close();

Document document = new Document();

PdfWriter.getInstance(document, new FileOutputStream("fileHistory.pdf"));

Font f1 = FontFactory.getFont("DejaVuSans.ttf", "cp1251", BaseFont.EMBEDDED, 10);

document.open();

ArrayList<String> name = null;

try {

name = DB.getHist();

} catch (SQLException e) {

throw new RuntimeException(e);

}

for(int i = 0; i < name.size(); i++){

List list = new List();

list.add((new ListItem("FIO: "+ name.get(i) +"\t", f1)));

list.add((new ListItem("Last Enter: "+name.get(++i)+"\t\n\n", f1)));

document.add(list);

}

document.close();

statusOtchet.setVisible(true);

}

Решение: Добавить строчки для шрифтов, добавляющий русский символы.

Font f1 = FontFactory.getFont("DejaVuSans.ttf", "cp1251", BaseFont.EMBEDDED, 10);

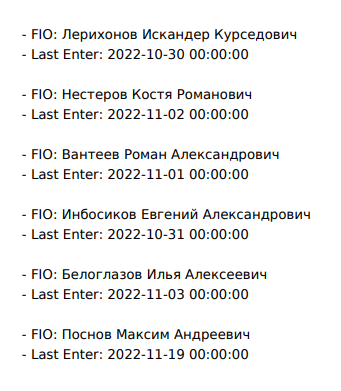


Рис. 20. Отчет по разделу «История входа»

1. **Руководство пользователя**
   1. **Страница авторизации пользователя**

Для того, чтобы попасть в личный кабинет и получить доступ ко всему ему функционалу, необходимо совершить вход в систему посредством авторизации на форме приложения.

На форме будет предложенно ввести логин и пароль.

После ввода персональных данных необходимо нажать на кнопку «Войти» и, в случае ввода корректных данных пользователь получает доступ к Личному кабинету сотрудника, в противном случае на форму будет выдано предупреждение: «Неверный логин или пароль!».

1. Вводим логин и пароль, далее нажимаем на кнопку «Войти» (см. рис. 3);
2. В случае успешной авторизации попадаем на профиль сотрудника (см. рис. 4, 5, 6);
   1. **Профиль сотрудника**

Находясь на экране «Профиль» сотрудник, в зависимости от его должности, может выбрать любую доступную ему функцию (см. рис. 4, 5, 6).

* 1. **Добавление новой услуги**

Находясь на экране «Профиль» сотрудник должен кликнуть по кнопке с названием «Добавить новую услугу», чтобы добавить новую услугу. После этого выполняется переход на экран «Добавление новой услуги».

Далее нужно ввести свободный ID услуги, название услуги и стомость услуги, далее нажать на кнопку «Добавить». В случае успешного добавления услуги выведется надпись «Услуга добавлена», в случае провала будет информация об ошибке.

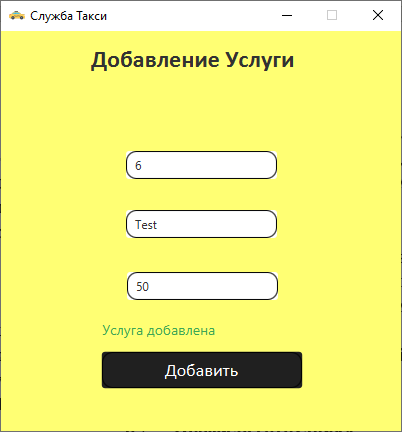


Рис. 21. Добавление новой услуги

* 1. **Регистрация нового клиента**

Находясь на экране «Профиль» сотрудник должен кликнуть по кнопке с названием «Регистрация нового клиента», чтобы добавить нового клиента. После этого выполняется переход на экран «Регистрация нового клиента».

Далее нужно ввести свободный ID клиента, ФИО, адрес, дату рождения, логин и пароль, далее нужно нажать на кнопку «Зарегистрировать». В случае успешного добавления услуги выведется надпись «Клиент зарегистрирован», в случае провала будет информация об ошибке.

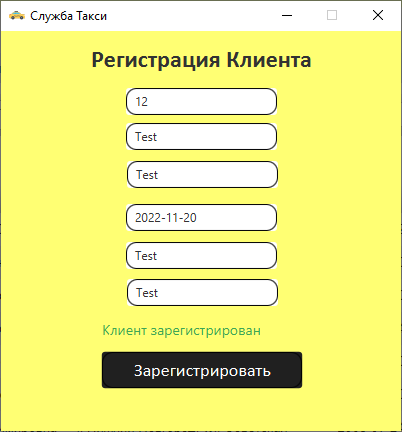


Рис. 22. Регистрация нового клиента

* 1. **Добавление нового работника**

Находясь на экране «Профиль» сотрудник должен кликнуть по кнопке с названием «Добавить нового работника», чтобы добавить нового работника. После этого выполняется переход на экран «Добавление работника».

Далее нужно ввести свободный ID работника, ФИО, должность, адрес, дату рождения, логин и пароль, далее нужно нажать на кнопку «Зарегистрировать». В случае успешного добавления услуги выведется надпись «Работник зарегистрирован», в случае провала будет информация об ошибке.

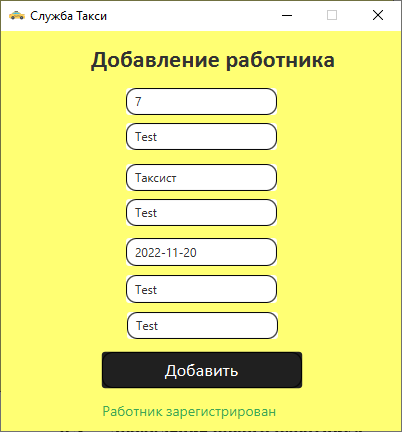


Рис. 23. Добавление нового работника

* 1. **Просмотр истории входа**

Находясь на экране «Профиль» сотрудник должен кликнуть по кнопке с названием «История входа», чтобы просмотреть работников по истории входа. После этого выполняется переход на экран «Просмотр работников по истории входа» (см. рис. 10).

На этом экране сотрудник может также сформировать и скачать отчет об истории входа сотрудников (см. рис. 11).

* 1. **Просмотр информации о работниках**

Находясь на экране «Профиль» сотрудник должен кликнуть по кнопке с названием «Просмотреть работников», чтобы просмотреть информацию о работниках службы такси. После этого выполняется переход на экран «Просмотр работников» (см. рис. 12).

На этом экране сотрудник может также сформировать и скачать отчет об информации сотрудников (см. рис. 13).

* 1. **Формирование заказа**

Находясь на экране «Профиль» сотрудник должен кликнуть по кнопке с названием «Сформировать заказ», чтобы сформировать заказ. После этого выполняется переход на экран «Формирование заказа».

На этом экране сотруднику нужно указать ID свободного заказа, дату заказа, время заказа, ID клиента, ID услуги и нажать на кнопку «Сформировать». В случае успешног формирования заказа, будет показана надпись «Заказ сформирован», в случае провала будет показана информации об ошибке.

Также на данном экране после формирования заказа, сотрудник может сформировать штрих-код, нажав на кнопку «Сформировать штрих-код». В последующем сотрудник может его скачать, нажам на кнопку «Скачать штрих-код» и выбрать пусть для сохранения файла pdf формата.

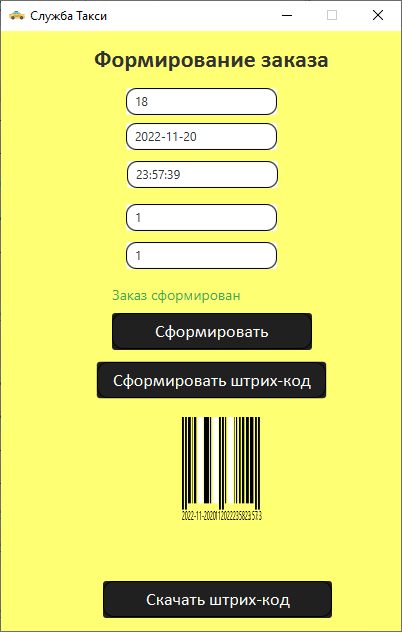


Рис. 24. Формирование заказа и формирование штрих-кода

* 1. **Просмотр информации о клиентах**

Находясь на экране «Профиль» сотрудник должен кликнуть по кнопке с названием «Просмотреть клиентов», чтобы просмотреть информацию о клиентах службы такси. После этого выполняется переход на экран «Просмотр клиентов» (см. рис. 16).

На этом экране сотрудник может также сформировать и скачать отчет об информации клиентов (см. рис. 17).

* 1. **Принятие заказа**

Находясь на экране «Профиль» сотрудник должен кликнуть по кнопке с названием «Принять заказ», чтобы выбрать невыполненные заказы. После этого выполняется переход на экран «Принять заказ».

На данном экране таксисты должны выбрать из выпадающего списка невыполненные заказы, выбрать заказ и, в зависимости от ситуации, завершить или отменить заказ, указав время поездки.

В случае завершения поездки, будет показана надпись «Завершено», в случае отмены поездки, будет показана надпись «Отменено».

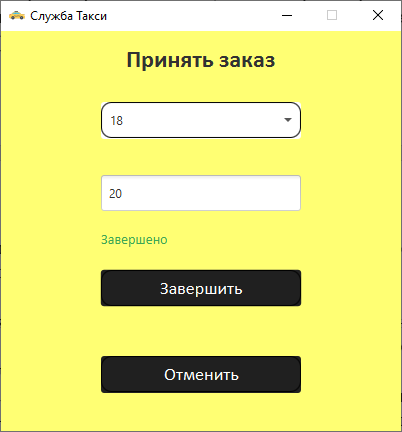


Рис. 25. Завершение заказа

1. **Заключение**

Итогом данного курсового проекта является приложение «UberNN» для одноименной службы такси.

Разработанный модуль позволяет вести полный контроль за своей рабочей деятельностью внутри службы такси, а также управлять пользовательскими данными клиентов, работников и заказами. Проект выполнен в полном соответствии с заданием на курсовое проектирование.

В проекте разработана эксплуатационная документация, для подробного ознакомления пользователей с работой в системе.

Таким образом, можно сделать вывод, что во время разработки были достигнуты все цели создания системы и, что её внедрение должно привести к повышению производительности и качества службы такси.

**Список использованных источников**

1. Машнин Т. С.JavaFX 2.0: разработка RIA-приложении?. / Тимур Машнин. - СПб:БХВ-Петербург, 2012. - 320 с.
2. Обзор JavaFXSceneBuilder[Электронный ресурс] - Режим доступа: http://docs.oracle.com/javafx/scenebuilder/1/overview/jsbpub-overview.htm
3. Учебник по JavaFX 8 [Электронный ресурс] - Режим доступа: http://code.makery.ch/library/javafx-8-tutorial/ru/
4. JavaFX [Электронный ресурс] - Режим доступа: https://ru.wikipedia.org/wiki/JavaFX