Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Программирование сетевых приложений»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Ассистент кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.П. Лыщик |
|  | \_\_\_.\_\_\_\_.2022 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«**Программа по учету и комплектации фирм информационными технологиями (ПО)**»**

БГУИР КР 1-40 01 02-08 005 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 073601  Гайсенок Данила Павлович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2022  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2022

Содержание

[ВВЕДЕНИЕ 5](#_Toc121796922)

[1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА 6](#_Toc121796923)

[1.1 Описание предметной области 6](#_Toc121796924)

[1.2 Разработка функциональной модели предметной области 7](#_Toc121796925)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 9](#_Toc121796926)

[1.4 Разработка информационной модели предметной области 10](#_Toc121796927)

[1.5 Модели представления программного средства и их описание 12](#_Toc121796928)

[2 КОНСТРУИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 15](#_Toc121796929)

[2.1 Постановка задачи 15](#_Toc121796930)

[2.2 Архитектурные решения 15](#_Toc121796931)

[2.3 Описание алгоритмов, реализующих бизнес логику разрабатываемого программного средства 17](#_Toc121796932)

[2.4 Проектирование пользовательского интерфейса 21](#_Toc121796933)

[2.5 Обоснование выбора компонентов и технологий для реализации программного средства 24](#_Toc121796934)

[3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА 26](#_Toc121796935)

[4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРИЛОЖЕНИЯ 32](#_Toc121796936)

[ЗАКЛЮЧЕНИЕ 44](#_Toc121796937)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45](#_Toc121796938)

[ПРИЛОЖЕНИЕ А 46](#_Toc121796939)

[ПРИЛОЖЕНИЕ Б 47](#_Toc121796940)

[ПРИЛОЖЕНИЕ В 62](#_Toc121796941)

[ПРИЛОЖЕНИЕ Г 67](#_Toc121796942)

# ВВЕДЕНИЕ

На сегодняшний день информационные технологии играют очень важную роль для каждого человека. Любая сфера жизни людей зависит от компьютеров, мобильных телефонов, современного ПО и т.д. Эти устройства и программы позволяют общаться на больших расстояниях, обмениваться фото- и видео сообщениями, текстовыми и другими файлами.

Очень легко отслеживать и вести учет записей и бизнес-процессов на любом предприятии благодаря современным технологиям. Они позволяют автоматизировать и ускорить процесс работы с информацией, что является актуально в век высоких технологий, ведь самому человеку было бы трудно фиксировать данные и сортировать их вручную.

Так, например, компания (фирма) которая представляет из себя огромный организм состоящий из различных процессов и задач требует полной автоматизации. Например, контроль лицензий на программное обеспечение и наличие необходимого, для обеспечения полноценной функциональности в решении рабочих задач, программного обеспечения на всех вычислительных машинах.

Автоматизация процесса учета и комплектации информационными технологиями позволяет решать организационные вопросы, более оперативно реагировать на подходящий к завершению срок лицензии на программное обеспечение, усилить контроль за работоспособностью закупленных вычислительных машин.

При увеличении уровня автоматизации на этапе контроля за лицензиями на установленное программное обеспечение появляется возможность более точно прогнозировать время следующего продления лицензии на закупленное программное обеспечение, рассчитать траты на продление конкретного количества лицензий для вычислительных машин фирмы.

Объектом исследования является процесс автоматизации учета и комплектации фирм информационными технологиями, предмет данного исследования – создание системы по учету и комплектации информационными технологиями(ПО).

# 1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА

* 1. **Описание предметной области**

Управление ИТ активами на предприятии включает в себя два этапа:

* Технический

Управление операциями с ИТ-активами на протяжении всего их жизненного цикла, от приобретения до вывода из эксплуатации: планирование приобретения ИТ-активов, установка оборудования и ПО, отслеживание всех происходящих с ними изменений, интеграция со службами поддержки для своевременного устранения неисправностей и списание или замена ИТ-актива по истечении отведенного ему срока эксплуатации. [1]

* Финансовый

Сравнительный анализ затрат на покупку и взятие в аренду, управление контрактами на приобретение или покупку активов, определение и оптимизация совокупной стоимости владения активом (total cost of ownership, TCO) на протяжении его жизненного цикла. [1]

Управление ИТ активами это один из основных компонентов эффективной работы ИТ фирмы. Отлаженный процесс управления этими активами с добавлением автоматизированной системы позволяет обеспечить полное представление о состоянии ИТ активов и связанных с ними затрат, позволяют решить ряд проблем:

* + 1. Сбор и анализ информации об ИТ активах – позволяют в автоматическом режиме собирать данные об их использовании и взаимосвязи с оборудованием.
    2. Ввод в эксплуатацию нового вида актива – можно подавать запросы на изменение услуг и активов
    3. Управление потреблением и распределением активов – упорядочивает работу с активами, распределение по логическим группам
    4. Оптимизация процесса обновления ИТ-активов
    5. Координация ремонтов и модернизаций

Автоматизация процесса учета и комплектации информационными технологиями позволяет решать организационные вопросы, более оперативно реагировать на подходящий к завершению срок лицензии на программное обеспечение, усилить контроль за работоспособностью закупленных вычислительных машин.

При увеличении уровня автоматизации на этапе контроля за лицензиями на установленное программное обеспечение появляется возможность более точно прогнозировать время следующего продления лицензии на закупленное программное обеспечение, рассчитать траты на продление конкретного количества лицензий для вычислительных машин фирмы.

Данные системы позиционируются как системы, которые, помимо решения задач планирования ресурсов также могут решать вопросы организации данных и оптимизации внутренних процессов.

* 1. **Разработка функциональной модели предметной области**

Процесс работы автоматизированной системы по учету и комплектации фирм ИТ можно разделить на следующие этапы:

* Получение данных от клиента
* Приведение данных и формирование сводных таблиц
* Продление лицензии или приобретение новой и оплата
* Обновление данных и разработка дальнейших рекомендаций по работе с ИТ-активами

Рассматриваемые в данном курсовом проекте бизнес-процессы поликлиники, связанные с обслуживанием пациентов, построением отчетов и принятием управленческих решений, можно представить в виде системы, спроектированной в *CASE*-средстве *AllFusion* *Process* *Modeler* с точки зрения стандарта *IDEF*0.

В качестве входных параметров выступают данные о клиенте и об оказываемых услугах. Управляющее воздействие оказывают сроки лицензий и количество рабочих станций. Механизм осуществления составляют сотрудник компании и программное обеспечение по учету и комплектации фирмы ИТ. Выходным процессом является автоматизированный процесс учета и комплектации фирм ИТ. Функциональная модель представлена на рисунках 1.1 – 1.4

|  |
| --- |
|  |
| Рисунок 1.1 – Верхний уровень функциональной модели |
|  |
| Рисунок 1.2 – Декомпозиция контекстной диаграммы |
|  |
| Рисунок 1.3 – Декомпозиция блока «Сформировать сроки службы и сводные таблицы» |
|  |
| Рисунок 1.4 – Декомпозиция блока «Управлять продлением и приобретением новых лицензий/АО» |

Перед проектированием программного средства следует определить, какие требования должны быть предъявлены к нему, так как невыполнение некоторых требований, которые были сформированы еще на ранней стадии, говорит о том, что разработанный сервис не сможет эффективно и с максимальной пользой использоваться, как было задумано изначально.[3]

* 1. **Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

Для разработки программного средства необходимо понять принцип работы по учету и комплектации фирм ИТ.

Так как каждая фирма использует в своей деятельности разное количество рабочих станций, на которых необходимо установить разное и специфическое программное обеспечение для эффективной работы, то необходимо чтобы для каждой фирмы в отдельности была возможность в отдельности управлять и контролировать состояние каждой лицензии и рабочей станции.

Для хранения информации необходимо создать базу данных, в которой будет содержаться вся информация о фирмах, лицензиях и рабочих станциях. Для этого будет использован MySQL Server. Подключение к нему будет осуществляться через сервер, на котором будут обрабатываться запросы от пользователей.

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью вариантов использования. Диаграмма вариантов использования программного средства, реализованного в данном проекте, представлена на рисунке 1.5

|  |
| --- |
|  |
| Рисунок 1.5 – Диаграмма вариантов использования |

Для взаимодействия с программным средством существует 2 роли: Администратор и пользователь. Набор предоставленных им возможностей отличается.

Администратор имеет полный доступ к информации и может добавлять, удалять и редактировать данные пользователей.

Пользователь может добавлять данные о своей фирме(ах), покупать, продлевать или отказываться от лицензий.

* 1. **Разработка информационной модели предметной области**

При проектировании системы было принято решение использовать следующие сущности

* Users
* Software
* Hardware
* Licenses
* Hwstatus
* Company
* Orders

Графическое отображение информационной модели представлено на рисунке 1.6

|  |
| --- |
|  |
| Рисунок 1.6 – Информационная модель |

Ниже приведено подробное описание атрибутов каждой сущности, входящей в модель.

Сущность Users содержит атрибуты:

idПользователя – для хранения уникального идентификатора;

Логин – логин пользователя, используется для входа в программу;

Пароль – Пароль пользователя, используется для входа в программу;

Сущность Company наследуется от сущности Users и содержит атрибуты:

idКомпании – уникальный идентификатор компании;

Название\_Компании – хранит название компании;

Количество\_рабочих\_станций – хранит количество рабочих станций компании;

idПользователя – унаследованный от сущности Users атрибут;

Сущность Licenses наследуется от сущностей Company и Software, содержит атрибуты:

id\_Лицензии - уникальный идентификатор;

id\_Компании – унаследованный от сущности Company атрибут;

id\_ПО – унаследованный от сущности Software атрибут;

Истечение\_Срока – хранит дату окончания лицензии;

Сущность Software содержит атрибуты:

Id\_ПО – уникальный идентификатор;

Название\_ПО – хранить название ПО;

Цена\_за\_год – хранит цену продления лицензии на это ПО на 1 год;

Производитель\_ПО – хранит название производителя ПО;

Сущность Hardware содержит атрибуты:

id\_АО – уникальный идентификатор;

Название\_АО – хранит название аппаратного обеспечения;

Цена\_АО – хранит цену аппаратного обеспечения;

Производитель\_АО – хранит название производителя аппаратного обеспечения;

Сущность HWstatus наследуется от сущностей Company и Hardware, содержит атрибуты:

id – уникальный идентификатор;

id\_Компании – унаследованный от сущности Company атрибут;

id\_АО – унаследованный от сущности Hardware атрибут;

Дата\_Покупки\_АО – дата покупки аппаратного обеспечения;

Дата\_Окончания\_Срока\_Эксплуатации – дата окончания срока эксплуатации аппаратного обеспечения;

Количество – количество рабочих станций в компании;

Сущность Orders наследуется от сущностей Company, Hardware и содержит атрибуты:

id-Заказа – уникальный идентификатор;

id\_Компании – унаследованный от сущности Company атрибут;

id\_АО – унаследованный от сущности Hardware атрибут;

Количество – хранит количество заказанного АО;

Одобрено – заполняется администратором, который решает одобрить ли заказ.

*SQL*-скрипт для генерации базы данных приведен в приложении В.

* 1. **Модели представления программного средства и их описание**

**1.5.1** Диаграмма состояний

Для  моделирования  поведения  на  логическом  уровне  в  языке  UML могут использоваться сразу несколько канонических диаграмм: состояний, деятельности,  последовательности  и  кооперации,  каждая  из  которых фиксирует внимание на отдельном аспекте функционирования системы. [10]

Диаграмма состояний. Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние. Диаграммы состояний могут быть вложены друг в друга, образуя вложенные диаграммы более детального представления отдельных элементов модели. Для понимания семантики конкретной диаграммы состояний необходимо представлять не только особенности поведения моделируемой сущности, но и знать общие сведения по теории автоматов. [2, 4]

|  |
| --- |
| D:\Универ\ПСП\Курсовой\Диаграмма.jpg |
| Рисунок 1.7 – Диаграмма состояний |

**1.5.2** Диаграмма последовательности

|  |
| --- |
|  |
| Рисунок 1.8 Диаграмма последовательности |

На диаграмме последовательности показан процесс авторизации пользователей. Когда пользователь посылает логин и пароль на сервер для авторизации, сервер принимает данные, отправляет запрос к БД на выборку данных о пользователях. Получив данные о пользователях сервер проверяет на совпадение с полученными от клиента логином и паролем. При совпадении сервер отправляет ответ, что авторизация успешна. После этого отправляется запрос на предоставление пользовательского интерфейса.

**1.5.3** Диаграмма развёртывания

На диаграмме отражено, что для ПК пользователя обязательно наличие среды выполнения *jdk* 1.8+, также, как и для ПК, который будет выступать в качестве сервера. Взаимодействие между приложением клиента и сервера будет осуществляться посредством *TCP*/*IP* соединения. Для соединения с базой данных необходимо подключение *JDBC* *Driver*. База данных хранится на *MySQL* *SERVER*.

|  |
| --- |
|  |
| Рисунок 1.9 Диаграмма развёртывания |

1. КОНСТРУИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА
   1. Постановка задачи

Для разработки программного средства необходимо понять принцип работы по учету и комплектации фирм ИТ.

Так как каждая фирма использует в своей деятельности разное количество рабочих станций, на которых необходимо установить разное и специфическое программное обеспечение для эффективной работы, то необходимо чтобы для каждой фирмы в отдельности была возможность в отдельности управлять и контролировать состояние каждой лицензии и рабочей станции.

Для решения этих задач необходимо создать базу данных, в которой будет содержаться информация об пользователях, компаниях, оборудовании, заказах, лицензиях в электронном виде. Это позволит облегчить поиск и работу с данными для пользователя и администратора.

Для корректной работы необходимо предусмотреть возможность добавления, удаления и редактирования информации в базе данных. Для хранения информации будет использован *MYSQL* *Server*. Подключение к нему будет осуществляться при авторизации и в зависимости от полученных прав, пользователь будет получать определённый набор возможностей для выполнения своей работы.

Администратор будет отвечать за регистрацию и работу с данными пользователей, также будет работать с ПО, на которое будут продаваться лицензии:

* Добавление
* Удаление
* Редактирование

Пользователю необходимо работать с фирмой, представителем которой он является, видеть доступные лицензии, и срок их действия. Также он может просматривать и вводить данные о аппаратном обеспечении, используемом в его фирме.

* 1. Архитектурные решения

Диаграммы классов используются при моделировании программных средств наиболее часто. На диаграммах классов показываются классы, интерфейсы и отношения между ними.

В данном курсовом проекте используются 2 пакета с классами:

Client – в этом пакте содержатся 28 классов для реализации пользовательского интерфейса, часть из которых наследуется от класса JFrame для реализации оконного приложения, другая часть наследуется от класса AbstractTableModel для реализации таблиц на экране.

Реализованы классы AdminFrame, UserFrame, HardwareFrame, SoftwareFrame, CompanyFrame, LicenseFrame, OrdersFrame. Эти классы представляют собой окна на которых расположены объекты для взаимодействия с пользователями.

AdminFrame используется как главное окно для администратора, содержит в себе таблицу с информацией о пользователях, а также кнопки для взаимодействия с данными в этой таблице (Добавить, Изменить и Удалить). Также в этом окне присутствуют кнопки для перехода в другие окна: «Программное обеспечение» - для перехода в окно работы с программным обеспечением, «Аппаратное обеспечение» - для работы с информацией об аппаратном обеспечении, «Заказы» - переход в окно для просмотра и одобрения заказов от клиентов, «Компании» - для просмотра зарегистрированных компаний.

Server – в этом пакете реализуются классы для работы с соединением TCP-IP чтобы пересылать сообщения от сервера к клиенту и обратно. Также присутствуют классы реализующие интерфейс Serializable для пересылки объектов с данными из базы данных клиенту для последующей обработки.

Диаграмма классов пакета Client описана в приложении Г.

|  |
| --- |
|  |
| Рисунок 2.1 Диаграмма классов пакета Server |

Диаграмма компонентов описывает особенности физического представления системы. Она позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный и исполняемый код. [6,7] Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними. Диаграмму можно увидеть на рисунке 2.2.

|  |
| --- |
|  |
| Рисунок 2.2 Диаграмма компонентов |

* 1. Описание алгоритмов, реализующих бизнес логику разрабатываемого программного средства

Для входа в систему необходимо пройти авторизацию на первом окне, после прохождения авторизации, в зависимости от прав доступа откроется окно администратора или пользователя. Администратор в данном алгоритме участвует в качестве механизма одобрения или отклонения заказа, после его создания. Клиент входит в программу через механизм авторизации, далее для заказа оборудования необходимо выбрать компанию(фирму), связанную с данным пользователем от имени которой будет производится заказ. Далее происходит выбор типа аппаратного обеспечения и его количество. После чего происходит отправка заказа на одобрение.

|  |
| --- |
|  |
| Рисунок 2.3 Укрупненная схема алгоритма |

Рассмотрим ветку работа администратора

После авторизации под ролью администратор открывается окно для выбора функции с которой будет производиться работа.

Для администратора доступны варианты функций для работы с пользователями (добавление, удаление, редактирование информации), аппаратным обеспечением и программным обеспечением (добавление, удаление, обновление информации), просмотр компаний, созданных пользователями и работа с заказами (будет рассмотрено далее). Работа администратора представлена на рисунке 2.4.

Также рассмотрим работу пользователя. Для пользователей в данной системе доступна работа с компаниями (добавление, редактирование и удаление информации о них), работа с АО (внесение информации о количестве аппаратного обеспечения в фирме, редактирование этой информации), работа с лицензиями на ПО (Внесение информации о наличии лицензии у фирмы, редактировании этой информации, покупка новых лицензий), работа с заказами (будет рассмотрено далее). Работа пользователя представлена на рисунке 2.5.

|  |
| --- |
|  |
| Рисунок 2.4 Блок-схема работы администратора |
|  |
| Рисунок 2.5 Работа пользователя |

Рассмотрим реализацию алгоритма составления заказа пользователем и одобрения заказа администратором.

Далее рассмотрим схему работы алгоритма с позиции администратора (см. рисунок 2.6)

Процесс входа в систему под ролью администратора был показан на рисунке 2.3. Работа администратора на данном участке заключается в том, что он переходит на экран с уже сформированными пользователями заказами, для их дальнейшего одобрения. Далее происходит выбор конкретного заказа, который будет одобрен или отклонен, после чего администратор принимает решение: Для одобрения вызывается диалоговое окно с информацией о заказе и кнопкой одобрить или отмена. При нажатии кнопки отмена заказ остаётся неодобренным, при нажатии кнопки одобрить статус заказа меняется.

|  |
| --- |
|  |
| Рисунок 2.6 Реализация алгоритма со стороны администратора |

Рассмотрим схему реализации алгоритма со стороны пользователя (см. рисунок 2.7)

Для оформления заказа пользователь, прошедший авторизацию (см. рисунок 2.3), выбирает фирму, от имени которой будет составляться заказ, далее, при нажатии на кнопку «Заказы», открывается диалоговое окно с возможностью выбора аппаратного обеспечения из предложенного. После вводится количество заказываемого оборудования и заказ отправляется на одобрение после нажатия кнопки «Отправить».

|  |
| --- |
|  |
| Рисунок 2.7 Реализация алгоритма со стороны пользователя |

* 1. Проектирование пользовательского интерфейса

Пользовательский интерфейс – это система средств для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т.п.). При этом, в отличие от интерфейса командной строки, пользователь имеет произвольный доступ (с помощью клавиатуры или указательного устройства ввода) ко всем видимым экранным объектам, а на экране реализуется модель мира в соответствии с некоторой метафорой и осуществляется прямое манипулирование.[9]

Приложение имеет графический интерфейс, реализованный на Java Swing.

Все компоненты взаимодействуют друг с другом и связаны. Так же могут работать по отдельности. Для каждой части разработана таблица в которой содержатся и хранятся данные.

|  |
| --- |
|  |
| Рисунок 2.8 Интерфейс Администратора |

Для администратора доступно управление пользователями путем модификации записей в таблице, изменения которой заносятся в базу данных. Также администратор может управлять записями о продаваемых лицензиях и аппаратном обеспечении.

Для пользователя доступно управление компаниями, от имени которых будут создаваться заказы, покупаться лицензии и т.д., с помощью внесения и изменения данных в таблице. Также доступна работа с лицензиями на программное обеспечение и работа с информацией об имеющемся в фирме аппаратным обеспечением. Интерфейс пользователя представлен на рисунке 2.9.

Для входа в систему необходимо пройти авторизацию. Окно авторизации открывается при первом запуске программы и меет следующий функционал:

Для входа используются два текстовых поля для ввода логина и пароля, для отправки данных на проверку используется кнопка «Войти».

По нажатию на кнопку «Регистрация» открывается диалоговое окно, в котором можно зарегистрировать нового пользователя. Интерфейс окна авторизации представлен на рисунке 2.10, окна регистрации – на рисунке 2.11.

|  |
| --- |
|  |
| Рисунок 2.9 Интерфейс пользователя |
|  |
| Рисунок 2.10 Интерфейс окна авторизации |
|  |
| Рисунок 2.11 Интерфейс окна регистрации |

* 1. Обоснование выбора компонентов и технологий для реализации программного средства

Программное приложение должно быть написано на объектно-ориентированном языке *Java* и использовать архитектуру «клиент-сервер». Клиент-серверная архитектура – взаимодействие двух самостоятельных процессов – клиента и сервера, которые могу выполняться как на одном, так и на разных компьютерах, обмениваясь данными по сети.

Серверы – программы, которые реализуют определённый функционал для обработки запросов клиента и связи с базами данных.

Клиенты – программы, которые реализуют пользовательский интерфейс для взаимодействия с сервером. Процесс взаимодействия с сервером происходит путем пересылки сообщений.

Таким образом, архитектура «клиент-сервер» позволяет эффективно распределить работу между клиентскими и серверными частями системы: приложения, которые работают на стороне клиента не читают записи базы данных «напрямую», а посылают запросы на сервер, где они обрабатываются, результаты обработки отсылаются назад клиенту, что сокращает потоки информации.

|  |
| --- |
| https://pandia.ru/text/80/491/images/img10_27.jpg |
| Рисунок 2.12 Архитектура клиент-серверного взаимодействия |

Клиент реализует пользовательский интерфейс. Для этого используется стандартные библиотеки пользовательского интерфейса *SWING*.

Бизнес-логика курсового проекта выполняется на серверной части. Клиент отправляет запросы на сервер, где с помощью *SQL*-запросов будет добавляться, обновляться, удаляться и выбираться вся необходимая информация из БД.

Основной средой разработки был выбран *IDE IntelliJ IDEA*. *MySQL Server* используется в качестве СУБД. *Enterprise Architect* используется для разработки и построения *UML*-диаграмм (*Use Case, Statechart, Sequence diagram*, диаграмма классов, *Component diagram, Deployment diagram*). *CASE-*средство *CA AllFusion Process Modeler r*7 (*BPwin*) используется для проектирование *IDEF*0 модели. *CA AllFusion ERwin Data Modeler r*7 (*ERwin*) используется для проектирования информационной модели.

3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Тестирование – это процесс выполнения программы, целью которого является выявление ошибок.

Функциональные спецификации программного средства приведены в разделе 1.3 и описываются вариантами использования.

Рассмотрим ситуацию некорректной авторизации пользователя:

* Ввод логина без ввода пароля (см. рисунок 3.1)
* Ввод пароля без ввода логина (см. рисунок 3.2)
* Попытка авторизироваться без ввода обоих полей (см. рисунок 3.3)
* Ввод неправильного логина или пароля (см. рисунок 3.4)

Программа реагирует на все ситуации и выдаёт пользователю предупреждение о некорректности ввода.

|  |
| --- |
|  |
| Рисунок 3.1 Попытка входа без ввода пароля |
|  |
| Рисунок 3.2 Попытка входа без ввода логина |
|  |
| Рисунок 3.3 Попытка входа без ввода полей |
|  |
| Рисунок 3.4 Попытка входа с неправильным логином или паролем |

Далее рассмотрим ситуации некорректного ввода при регистрации пользователя

|  |
| --- |
|  |
| Рисунок 3.5 Попытка регистрации с пустыми полями |

При регистрации пользователя срабатывают такие же предупреждения, как и при попытке авторизации:

* Попытка регистрации без ввода пароля (см. рисунок 3.6)
* Попытка регистрации без ввода логина (см. рисунок 3.7)
* Попытка регистрации без ввода обоих полей (см. рисунок 3.5)

|  |
| --- |
|  |
| Рисунок 3.6 Попытка регистрации без ввода пароля |
|  |
| Рисунок 3.7 Попытка регистрации без ввода логина |

Также исключительные ситуации, описанные для регистрации пользователя, работают и для добавления нового пользователя из роли администратора.

При изменении информации о пользователе от лица администратора происходит обработка ситуаций введения только логина и только пароля (см. рисунок 3.8-3.9)

Рассмотрим ситуации при попытке добавления нового ПО:

* Не введено ни одно из полей (см. рисунок 3.13)
* Не введено одно из полей (см. рисунок 3.10-3.12)

Аналогичным образом работает предупреждение о некорректном вводе при попытке изменить информацию об программном обеспечении (см. рисунок 3.14), добавить информацию об аппаратном обеспечении и редактировании информации об аппаратном обеспечении.

|  |
| --- |
|  |
| Рисунок 3.8 Введение только логина |
|  |
| Рисунок 3.9 Введение только пароля |
|  |
| Рисунок 3.10 Введение только названия |
|  |
| Рисунок 3.11 Введение только цены |
|  |
| Рисунок 3.12 Введение только производителя |
|  |
| Рисунок 3.13 Попытка добавления без ввода полей |
|  |
| Рисунок 3.14 Исключительные ситуации при изменении информации о ПО |

# 4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРИЛОЖЕНИЯ

В первую очередь необходимо запустить сервер проекта, для этого можно использовать файл Server.java. Для запуска программы необходимо открыть консоль, перейти в папку с файлом и ввести команду javac Server.java, после того как программа скомпилируется, вводится команда java Server и программа начинает работать.

Есть ещё один способ запустить программу – через IDE Intellij Idea, для чего необходимо открыть проект в среде разработки, и запустить файл Server.java.

После запуска сервера необходимо запустить клиентскую часть программы. Для этого необходимо запустить файл Client.java.

После запуска откроется окно авторизации (см. рисунок 4.1). Пройдем авторизацию как Администратор, после чего откроется окно работы администратора (см. рисунок 4.2).

|  |
| --- |
|  |
| Рисунок 4.1 Окно авторизации |

Для прохождения авторизации необходимо ввести логин и пароль, которые будут отправлены на сервер для прохождения валидации, после прохождения которой, в зависимости от прав доступа, откроется окно администратора или клиента.

|  |
| --- |
|  |
| Рисунок 4.2 Окно Администратора |

У администратора есть инструменты управления пользователями: добавление, удаление и изменение информации о них. Начнем с добавления информации о пользователях. Для этого необходимо нажать справа вверху на кнопку «Добавить пользователя». После нажатия на кнопку откроется диалоговое окно (см. рисунок 4.3), в котором нужно ввести данные о новом пользователе.

|  |
| --- |
|  |
| Рисунок 4.3 Диалоговое окно добавления пользователя |

После добавления пользователя, чтобы он отобразился в таблице, необходимо нажать кнопку «Обновить таблицу», которая находится внизу справа. После обновления таблицы новый пользователь добавится в конец (см. рисунок 4.4).

|  |
| --- |
|  |
| Рисунок 4.4 Обновленная таблица |

Работая с пользователем можно изменить о нем информацию: сменить логин или пароль (см. рисунок 4.5-4.6), или удалить запись о нем (см. рисунок 4.7-4.8).

|  |
| --- |
|  |
| Рисунок 4.5 Диалоговое окно изменения информации |
|  |
| Рисунок 4.6 Обновленная информация |
|  |
| Рисунок 4.7 Удаление пользователя |
|  |
| Рисунок 4.8 После удаления |

Для просмотра ПО на которое продаются лицензии, необходимо нажать на кнопку «Программное обеспечение», после чего откроется окно работы с ПО (см. рисунок 4.9). Функции добавления, удаления и изменения реализованы подобным управлению пользователями образом. Для примера добавим новое ПО (см. рисунок 4.10).

|  |
| --- |
|  |
| Рисунок 4.9 Окно работы с программным обеспечением |
|  |
| Рисунок 4.10 После добавления |

После добавления нового ПО добавим аппаратное обеспечение New\_Hardware ценой 1250 производителя New\_Manufacturer (см. рисунок 4.11). Далее вернемся и создадим нового пользователя, для которого будем регистрировать компанию (см. рисунок 4.12-4.13). Создав пользователя выйдем из окна администратора, для того, чтобы зайти в окно пользователя.

|  |
| --- |
|  |
| Рисунок 4.11 Добавление нового АО |
|  |
| Рисунок 4.12 Окно пользователя |
|  |
| Рисунок 4.13 Добавление компании |

После добавления новой компании (см. рисунок 4.14) перейдем к введению информации о лицензиях, принадлежащих нашей компании (см. рисунок 4.16). Для этого необходимо в таблице внизу слева выбрать компанию, затем нажать на кнопку «Внести лицензию» вверху справа и ввести данные (см. рисунок 4.15). Далее введём информацию о том, какое конкретно у нас оборудование и в каком количестве, для этого необходимо выбрать компанию и нажать на кнопку «Внести данные об АО» после чего ввести данные.

Для продления лицензии на ПО необходимо выбрать нужную лицензию и нажать кнопку «Продлить лицензию». Откроется диалоговое окно, в котором будет предложено выбрать на какое количество лет будет продлена лицензия (см. рисунок 4.17), затем ввести данные счёта с которого будет происходить оплата (см. рисунок 4.18).

|  |
| --- |
|  |
| Рисунок 4.14 После добавления |
|  |
| Рисунок 4.15 Ввод данных о лицензии |
|  |
| Рисунок 4.16 После добавления |
|  |
| Рисунок 4.17 Продление лицензии |
|  |
| Рисунок 4.18 Продление лицензии |

После продления лицензии отобразится изменившаяся дата (см. рисунок 4.19). Далее произведем заказ оборудования New\_Hardware в количестве 35 штук. (см. рисунок 4.20). Для этого необходимо выбрать компанию и нажать кнопку «Заказы».

|  |
| --- |
|  |
| 4.19 Продленная лицензия |
|  |
| 4.20 Заказ оборудования |

После заказа оборудования и обновления таблицы заказ появится, и у пользователя (см. рисунок 4.21), и у администратора (см. рисунок 4.22). Далее администратор одобряет или отклоняет заказ (см. рисунок 4.23). Завершив процедуру одобрения, статус заказа изменится, и у администратора, и у пользователя (см. рисунок 4.24).

|  |
| --- |
|  |
| Рисунок 4.21 Обновление заказа пользователя |
|  |
| Рисунок 4.22 Обновление заказа у администратора |
|  |
| Рисунок 4.23 Одобрение заказа |
|  |
| Рисунок 4.24 Изменившийся статус заказа |

ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта было разработано программное приложение, позволяющие оптимизировать работу по учету и комплектации фирм ПО. В нем предусмотрена система администрирования, а также проверка учетных данных при авторизации пользователей.

Приложение удовлетворяет основным характеристикам, которые были заявлены. Оно удобно в эксплуатации, целостно, конкретизировано в рамках заданной предметной области.

При разработке данного программного продукта была учтена логика пользователя, и интерфейс данной программы сделан удобным и понятным.

Выполнено проектирование и разработка программного средства планирования и оценки нагрузки персонала медицинского учреждения: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработана информационная модель системы, представленная в виде схемы базы данных; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование системы показавшее ее соответствие функциональным требованиям, поставленным в задании на разработку.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Издательство «Открытые системы» [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.osp.ru/os/2011/03/13008203

[2] Технология разработки ПО [Электронный ресурс]. – Электронные данные. – Режим доступа: https://technologiarpo.blogspot.com/p/blog-page\_16.html

[3] Дубейковский, В. И. Эффективное моделирование с AllFusion Process Modeler 4.1.4 и AllFusion PM / В. И. Дубейковский. – Москва: Нолидж, 2011. – 284 с.

[4] Леоненков, А. В. Самоучитель UML / А. В. Леоненков. – СПб.: BHV, 2014. – 304 с.

[5] Роб П., Коронел К. Системы баз данных: проектирование, реализация и управление – М: БХВ-Петербург, 2004. – 1040 с.

[6] [Бабич, А. В. Введение в UML. [Электронный ресурс]. – Режим доступа : http://www.intuit.ru/studies/courses/1007/229/info.](file:///C:\Users\Svetlana\Downloads\Бабич%20А. В. Введение%20в%20UML.%20%5bЭлектронный%20ресурс%5d.%20–%20Электронные%20данные.%20Лекции.%20–%20Режим%20доступа:%20http:\www.intuit.ru\studies\courses\1007\229\info)

[7] Басс, Л., Архитектура программного обеспечения на практике. 2-е издание / Л. Басс, П. Клементс, Р. Кацман. – СПб.: Питер, 2006. – 575 с.

[8] Харрингтон Джен Л. Проектирование реляционных баз данных – М: Лори, 2016. – 514 с.

[9] Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер. – М.: Вильямс, 2016 – 544 с.

[10] Маклаков, С. В. Книга моделирования бизнес процессов при помощи BpWin 4.0: справ. пособие / С. В. Маклов. – Москва: Диалог-Мифи, 2002. – 74 с.

ПРИЛОЖЕНИЕ А

(обязательное)

Отчет о проверке на заимствования в системе «Атиплагиат»

|  |
| --- |
|  |
| Рисунок А.1 Отчет о проверке на заимствования в системе «Антиплагиат» |

# ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг кода алгоритмов, реализующих бизнес логику

Файл Server.java

package server;  
  
import java.sql.\*;  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
  
public class Server {  
 static int countClients = 0;  
 private static ObjectInputStream ois;  
 private static ObjectOutputStream oos;  
  
 public static void main(String[] args) {  
 String everything="";  
 try(BufferedReader br = new BufferedReader(new FileReader("file.txt"))) {  
 StringBuilder sb = new StringBuilder();  
 String line = br.readLine();  
  
 while (line != null) {  
 sb.append(line);  
 sb.append(System.lineSeparator());  
 line = br.readLine();  
 }  
 everything = sb.toString();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 String[] val = everything.split(" ");  
 String con = val[0];  
 //String con = "jdbc:mysql://localhost:3306/KP";  
 try (Connection connection = DriverManager.getConnection(con, "root", "Admin2001");) {  
 System.out.println("Connected");  
  
 ServerSocket sock = null;  
 DataInputStream is = null;  
 DataOutputStream os = null;  
 try {

Продолжение приложения Б

sock = new ServerSocket(1024);  
 while (true) {  
 Socket client = sock.accept();  
 countClients++;  
 System.out.println("==============================");  
 System.out.println("Client " + countClients + " connected");  
  
 //is = new DataInputStream( client.getInputStream());  
 //os = new DataOutputStream( client.getOutputStream());  
 ois = new ObjectInputStream(client.getInputStream());  
 oos = new ObjectOutputStream(client.getOutputStream());  
  
 ClientHandler clientSock = new ClientHandler(client, connection, oos, ois);  
 new Thread(clientSock).start();  
 }  
 } catch (IOException e) {  
 System.out.println("Disconnected");  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 //System.out.println("Disconnected");  
 }  
 }  
}

Файл ClientHandler.java

package server;  
  
import com.mysql.cj.protocol.Resultset;  
  
import java.io.\*;  
import java.net.Socket;  
import java.sql.\*;  
public class ClientHandler implements Runnable{  
 private static Socket clientSocket;  
 private static ObjectInputStream ois;  
 private static ObjectOutputStream oos;  
 private static Connection connection;  
 private static ResultSet rs;  
 private static Statement stmt;  
  
 public ClientHandler(Socket client, Connection connection,

Продолжение приложения Б

ObjectOutputStream oos, ObjectInputStream ois) {  
 this.clientSocket = client;  
 //this.is = is;  
 //this.os = os;  
 this.connection = connection;  
 this.oos = oos;  
 this.ois = ois;  
 try {  
 stmt = connection.createStatement();  
 }  
 catch (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void run() {  
 try{  
 Integer num = 0;  
 while (true/\*(line=ois.readUTF())!=null\*/) {  
 System.out.println("Ожидание");  
 String line=ois.readUTF();  
 //System.out.println(line);  
 num = Integer.parseInt(line);  
 System.out.println(num);  
 switch (num) {  
 case 1/\*Вход\*/: {  
 line = ois.readUTF();  
 System.out.println("Sent from client " + line);  
 String val[] = line.split(" ");  
 String query = "select count(\*) from users";  
 rs = stmt.executeQuery(query);  
 int number = 0;  
 while (rs.next()) {  
 number = rs.getInt(1);  
 }  
 query = "SELECT \* FROM users";  
 rs = stmt.executeQuery(query);  
 int j = 0;  
 Integer result = 0;  
 Integer[] id = new Integer[number];  
 String[] login = new String[number];  
 String[] pass = new String[number];  
 while (rs.next()) {

Продолжение приложения Б

id[j] = rs.getInt(1);  
 login[j] = rs.getString(2);  
 pass[j] = rs.getString(3);  
 j++;  
 }  
 int counter = 0;  
 for (int i = 0; i < login.length; i++) {  
 if (login[i].equals(val[0])) {  
 if (pass[i].equals(val[1])) {  
 result = 1;  
 System.out.println(result.toString());  
 oos.writeUTF(result.toString());  
 oos.flush();  
 if (id[i] == 1) {  
 oos.writeUTF("A");  
 oos.flush();  
 i = login.length;  
 break;  
 } else {  
 oos.writeUTF("U");  
 oos.flush();  
 oos.writeUTF(id[i].toString());  
 oos.flush();  
 i = login.length;  
 }  
 }  
 } else {  
 counter++;  
 //System.out.println(result.toString());  
 //os.writeUTF("0");  
 //os.flush();  
 }  
 if (counter == login.length) {  
 oos.writeUTF(result.toString());  
 oos.flush();  
 }  
 }  
 System.out.println("Конец 1");  
 }break;  
 case 2:/\*Регистрация\*/ {  
 while((line= ois.readUTF())!=null) {  
 System.out.println("Sent from client " + line);  
 String val[] = line.split(" ");  
 String query = "INSERT Users(Логин, Пароль) VALUES (?,?)";

Продолжение приложения Б

PreparedStatement statement = connection.prepareStatement(query);  
 statement.setString(1, val[0]);  
 statement.setString(2, val[1]);  
 statement.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 break;  
 }  
 System.out.println("Конец 2");  
 }break;  
 case 3/\*Данные в таблицу\*/:{  
 String query = "select count(\*) from users";  
 rs = stmt.executeQuery(query);  
 Integer number = 0;  
 while (rs.next()) {  
 number = rs.getInt(1);  
 }  
 oos.writeObject(number);  
 oos.flush();  
 query = "select \* from users";  
 rs = stmt.executeQuery(query);  
 User user = new User(rs);  
 oos.writeObject((User)user);  
 oos.flush();  
 System.out.println("Конец 3");  
 }break;  
 case 4/\*Удаление пользователя\*/: {  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 String query = "DELETE FROM users WHERE idПользователя="+val[0];  
 stmt.executeUpdate(query);  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 4");  
 }break;  
 case 5/\*Изменение пользователя\*/: {  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 PreparedStatement prep = connection.prepareStatement("UPDATE users SET Логин=?, Пароль=? WHERE idПользователя=?");  
 prep.setString(1,val[1]);  
 prep.setString(2,val[2]);

Продолжение приложения Б

prep.setString(3,val[0]);  
 prep.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 5");  
 }break;  
 case 6/\*Заполнение таблицы ПО\*/:{  
 String query = "select count(\*) from software";  
 rs = stmt.executeQuery(query);  
 Integer number = 0;  
 while (rs.next()) {  
 number = rs.getInt(1);  
 }  
 oos.writeObject(number);  
 oos.flush();  
 query = "select \* from software";  
 rs = stmt.executeQuery(query);  
 Software sw = new Software(rs);  
 oos.writeObject((Software)sw);  
 oos.flush();  
 System.out.println("Конец 6");  
 }break;  
 case 7/\*Добавление ПО\*/:{  
 line = ois.readUTF();  
 System.out.println("Sent from client " + line);  
 String[] val = line.split(" ");  
 String query = "INSERT Software(Название\_ПО, Цена\_за\_год, Производитель\_ПО) VALUES(?,?,?)";  
 PreparedStatement statement = connection.prepareStatement(query);  
 statement.setString(1, val[0]);  
 statement.setString(2, val[1]);  
 statement.setString(3,val[2]);  
 statement.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 7");  
 }break;  
 case 8/\*Удаление По\*/:{  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 String query = "DELETE FROM software WHERE id\_ПО="+val[0];  
 stmt.executeUpdate(query);

Продолжение приложения Б

oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 8");  
 }break;  
 case 9/\*Редактирование ПО\*/:{  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 PreparedStatement prep = connection.prepareStatement("UPDATE software SET Название\_ПО=?, Цена\_за\_год=?, Производитель\_ПО=? WHERE id\_ПО=?");  
 prep.setString(1,val[0]);  
 prep.setString(2,val[1]);  
 prep.setString(3,val[2]);  
 prep.setString(4,val[3]);  
 prep.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 9");  
 }break;  
 case 10/\*Заполнение таблицы AО\*/:{  
 String query = "select count(\*) from hardware";  
 rs = stmt.executeQuery(query);  
 Integer number = 0;  
 while (rs.next()) {  
 number = rs.getInt(1);  
 }  
 oos.writeObject(number);  
 oos.flush();  
 query = "select \* from hardware";  
 rs = stmt.executeQuery(query);  
 Hardware hw = new Hardware(rs);  
 oos.writeObject((Hardware)hw);  
 oos.flush();  
 System.out.println("Конец 10");  
 }break;  
 case 11/\*Добавление AО\*/:{  
 line = ois.readUTF();  
 System.out.println("Sent from client " + line);  
 String[] val = line.split(" ");  
 String query = "INSERT Hardware(Название\_АО, Цена\_АО, Производитель\_АО) VALUES(?,?,?)";  
 PreparedStatement statement = connection.prepareStatement(query);  
 statement.setString(1, val[0]);

Продолжение приложения Б

statement.setString(2, val[1]);  
 statement.setString(3,val[2]);  
 statement.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 11");  
 }  
 case 12/\*Редактирование AО\*/:{  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 PreparedStatement prep = connection.prepareStatement("UPDATE hardware SET Название\_АО=?, Цена\_АО=?, Производитель\_АО=? WHERE id\_АО=?");  
 prep.setString(1,val[0]);  
 prep.setString(2,val[1]);  
 prep.setString(3,val[2]);  
 prep.setString(4,val[3]);  
 prep.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 12");  
 }  
 case 13/\*Удаление AO\*/: {  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 String query = "DELETE FROM hardware WHERE id\_АО="+val[0];  
 stmt.executeUpdate(query);  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 13");  
 }  
 case 14/\*Данные в таблицы Пользователя\*/: {  
 //license  
 String query = "select count(\*) from licenses";  
 rs = stmt.executeQuery(query);  
 Integer number = 0;  
 while (rs.next()) {  
 number = rs.getInt(1);  
 }  
 oos.writeObject(number);  
 oos.flush();  
 query = "select \* from licenses";  
 rs = stmt.executeQuery(query);

Продолжение приложения Б

License license = new License(rs);  
 oos.writeObject((License)license);  
 oos.flush();  
 //hwstatus  
 query = "select count(\*) from hwstatus";  
 rs = stmt.executeQuery(query);  
 number = 0;  
 while (rs.next()) {  
 number = rs.getInt(1);  
 }  
 oos.writeObject(number);  
 oos.flush();  
 query = "select \* from hwstatus";  
 rs = stmt.executeQuery(query);  
 Hws hws = new Hws(rs);  
 oos.writeObject((Hws)hws);  
 oos.flush();  
 //company  
 query = "select count(\*) from company";  
 rs = stmt.executeQuery(query);  
 number = 0;  
 while (rs.next()) {  
 number = rs.getInt(1);  
 }  
 oos.writeObject(number);  
 oos.flush();  
 query = "select \* from company";  
 rs = stmt.executeQuery(query);  
 Company com = new Company(rs);  
 oos.writeObject((Company)com);  
 oos.flush();  
 System.out.println("Конец 14");  
 }break;  
 case 15/\*Добавление компании\*/: {  
 line = ois.readUTF();  
 System.out.println("Sent from client " + line);  
 String[] val = line.split(" ");  
 String query = "INSERT Company(Название\_Компании, Количество\_рабочих\_станций, id\_Пользователя) VALUES(?,?,?)";  
 PreparedStatement statement = connection.prepareStatement(query);  
 statement.setString(1, val[0]);  
 statement.setString(2, val[1]);  
 statement.setString(3,val[2]);

Продолжение приложения Б

statement.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 15");  
 }break;  
 case 16/\*Добавление лицензии\*/:{  
 line = ois.readUTF();  
 System.out.println("Sent from client " + line);  
 String[] val = line.split(" ");  
 String query = "INSERT Licenses(id\_Компании, id\_По, Истечение\_Срока) VALUES(?,?,?)";  
 PreparedStatement statement = connection.prepareStatement(query);  
 statement.setString(1, val[0]);  
 statement.setString(2, val[1]);  
 statement.setString(3,val[2]);  
 statement.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 16");  
 }break;  
 case 17/\*Добавление лицензии АО\*/:{  
 line = ois.readUTF();  
 System.out.println("Sent from client " + line);  
 String[] val = line.split(" ");  
 String query = "INSERT Hwstatus(id\_Компании, id\_АО, Дата\_Покупки\_АО, Дата\_Окончания\_Срока\_Эксплуатации, Количество) VALUES(?,?,?,?,?)";  
 PreparedStatement statement = connection.prepareStatement(query);  
 statement.setString(1, val[0]);  
 statement.setString(2, val[1]);  
 statement.setString(3,val[2]);  
 statement.setString(4,val[3]);  
 statement.setString(5,val[4]);  
 statement.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 17");  
 }break;  
 case 18/\*Продление лицензии\*/:{  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 PreparedStatement prep = connection.prepareStatement("UPDATE

Продолжение приложения Б

licenses SET Истечение\_Срока= DATE\_ADD(Истечение\_Срока, INTERVAL ? YEAR) WHERE id\_Лицензии=?");  
 prep.setString(1,val[0]);  
 prep.setString(2,val[1]);  
 prep.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 18");  
 }break;  
 case 19/\*Удаление лицензии\*/:{  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 String query = "DELETE FROM licenses WHERE id\_Лицензии="+val[0];  
 stmt.executeUpdate(query);  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 19");  
 }break;  
 case 20/\*Редактирование лицензии АО\*/:{  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 PreparedStatement prep = connection.prepareStatement("UPDATE hwstatus SET id\_Компании=?, id\_АО=?, Дата\_Покупки\_АО=?, Дата\_Окончания\_Срока\_Эксплуатации=?,Количество=? WHERE id=?");  
 prep.setString(1,val[0]);  
 prep.setString(2,val[1]);  
 prep.setString(3,val[2]);  
 prep.setString(4,val[3]);  
 prep.setString(5,val[4]);  
 prep.setString(6,val[5]);  
 prep.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 20");  
 }  
 case 21/\*Удаление лицензии АО\*/:{  
 line = ois.readUTF();  
 //String[] val = line.split(" ");  
 String query = "DELETE FROM hwstatus WHERE id="+line;  
 stmt.executeUpdate(query);  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 21");

Продолжение приложения Б

}break;  
 case 22/\*Редактирование компании\*/:{  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 PreparedStatement prep = connection.prepareStatement("UPDATE company SET Название\_Компании=?, Количество\_рабочих\_станций=? WHERE idКомпании=?");  
 prep.setString(1,val[0]);  
 prep.setString(2,val[1]);  
 prep.setString(3,val[2]);  
 prep.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 22");  
 }break;  
 case 23/\*Удаление компании\*/:{  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 String query = "DELETE FROM hwstatus WHERE id\_Компании="+val[2];  
 stmt.executeUpdate(query);  
 query = "DELETE FROM licenses WHERE id\_Компании="+val[2];  
 stmt.executeUpdate(query);  
 query = "DELETE FROM company WHERE idКомпании="+val[2];  
 stmt.executeUpdate(query);  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 23");  
 }break;  
 case 24/\*Таблица заказы\*/:{  
 String query = "select count(\*) from orders";  
 rs = stmt.executeQuery(query);  
 Integer number = 0;  
 while (rs.next()) {  
 number = rs.getInt(1);  
 }  
 oos.writeObject(number);  
 oos.flush();  
 query = "select \* from orders";  
 rs = stmt.executeQuery(query);  
 //Orders or = new Orders(rs);  
 Orders or = new Orders(rs);

Продолжение приложения Б

oos.writeObject((Orders)or);  
 oos.flush();  
 System.out.println("Конец 24");  
 }break;  
 case 25/\*Добавить заказ\*/:{  
 line = ois.readUTF();  
 System.out.println("Sent from client " + line);  
 String[] val = line.split(" ");  
 String query = "INSERT Orders(id\_Компании, id\_АО, Количество) VALUES(?,?,?)";  
 PreparedStatement statement = connection.prepareStatement(query);  
 statement.setString(1, val[0]);  
 statement.setString(2, val[1]);  
 statement.setString(3,val[2]);  
 statement.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 25");  
 }break;  
 case 26/\*Удалить заказ\*/:{  
 line = ois.readUTF();  
 System.out.println("Sent from client " + line);  
 String[] val = line.split(" ");  
 String query = "DELETE FROM Orders WHERE id\_Заказа="+val[0];  
 stmt.executeUpdate(query);  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 26");  
 }break;  
 case 27/\*Изменить заказ\*/:{  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 PreparedStatement prep = connection.prepareStatement("UPDATE Orders SET Количество=? WHERE id\_Заказа=?");  
 prep.setString(1,val[0]);  
 prep.setString(2,val[1]);  
 prep.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 27");  
 }break;  
 case 28/\*Инфа компаний\*/:{

Продолжение приложения Б

String query = "select count(\*) from company";  
 rs = stmt.executeQuery(query);  
 Integer number = 0;  
 while (rs.next()) {  
 number = rs.getInt(1);  
 }  
 oos.writeObject(number);  
 oos.flush();  
 query = "select \* from company";  
 rs = stmt.executeQuery(query);  
 Company com = new Company(rs);  
 oos.writeObject((Company)com);  
 oos.flush();  
 System.out.println("Конец 28");  
 }break;  
 case 29/\*добрить заказ\*/:{  
 line = ois.readUTF();  
 String[] val = line.split(" ");  
 PreparedStatement prep = connection.prepareStatement("UPDATE Orders SET Одобрено=? WHERE id\_Заказа=?");  
 prep.setString(1,val[0]);  
 prep.setString(2,val[1]);  
 prep.execute();  
 oos.writeUTF("Command proceeded");  
 oos.flush();  
 System.out.println("Конец 29");  
 }break;  
 }  
 }  
  
 } catch (IOException e) {  
 System.out.println("Disconnected");  
 } catch (SQLException e) {  
 System.out.println("Disconnected");  
 } finally {  
 try{  
 stmt.close();  
 if (oos != null){  
 oos.close();  
 }  
 if (ois != null){  
 ois.close();  
 clientSocket.close();  
 }

Продолжение приложения Б

} catch (Exception e) {  
 System.out.println("Disconnected");  
 }  
 }  
 }  
}

# ПРИЛОЖЕНИЕ В

(обязательное)

Листинг скрипта генерации базы данных

-- MySQL Script generated by MySQL Workbench

-- Sun Dec 11 10:47:45 2022

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema KP

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema KP

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `KP` DEFAULT CHARACTER SET utf8 ;

USE `KP` ;

-- -----------------------------------------------------

-- Table `KP`.`Users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `KP`.`Users` (

`idПользователя` INT NOT NULL AUTO\_INCREMENT,

`Логин` VARCHAR(20) NOT NULL,

`Пароль` VARCHAR(20) NOT NULL,

PRIMARY KEY (`idПользователя`))

ENGINE = InnoDB;

Продолжение приложения В

-- -----------------------------------------------------

-- Table `KP`.`Company`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `KP`.`Company` (

`idКомпании` INT NOT NULL AUTO\_INCREMENT,

`Название\_Компании` VARCHAR(20) NOT NULL,

`Количество\_рабочих\_станций` INT NOT NULL,

`id\_Пользователя` INT NOT NULL,

PRIMARY KEY (`idКомпании`),

INDEX `id\_Пользователя\_idx` (`id\_Пользователя` ASC) VISIBLE,

CONSTRAINT `id\_Пользователя`

FOREIGN KEY (`id\_Пользователя`)

REFERENCES `KP`.`Users` (`idПользователя`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `KP`.`Software`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `KP`.`Software` (

`id\_ПО` INT NOT NULL AUTO\_INCREMENT,

`Название\_ПО` VARCHAR(20) NOT NULL,

`Цена\_за\_год` INT NOT NULL,

`Производитель\_ПО` VARCHAR(20) NOT NULL,

PRIMARY KEY (`id\_ПО`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `KP`.`Hardware`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `KP`.`Hardware` (

`id\_АО` INT NOT NULL AUTO\_INCREMENT,

`Название\_АО` VARCHAR(20) NOT NULL,

`Цена\_АО` VARCHAR(20) NOT NULL,

Продолжение приложения В

`Производитель\_АО` VARCHAR(20) NOT NULL,

PRIMARY KEY (`id\_АО`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `KP`.`Licenses`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `KP`.`Licenses` (

`id\_Лицензии` INT NOT NULL AUTO\_INCREMENT,

`id\_Компании` INT NOT NULL,

`id\_ПО` INT NOT NULL,

`Истечение\_срока` DATE NOT NULL,

PRIMARY KEY (`id\_Лицензии`),

INDEX `Licenses\_Software\_idx` (`id\_ПО` ASC) VISIBLE,

INDEX `Licenses\_Company\_idx` (`id\_Компании` ASC) VISIBLE,

CONSTRAINT `Licenses\_Software`

FOREIGN KEY (`id\_ПО`)

REFERENCES `KP`.`Software` (`id\_ПО`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `Licenses\_Company`

FOREIGN KEY (`id\_Компании`)

REFERENCES `KP`.`Company` (`idКомпании`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `KP`.`HWstatus`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `KP`.`HWstatus` (

`id` INT NOT NULL AUTO\_INCREMENT,

`id\_Компании` INT NOT NULL,

`id\_АО` INT NOT NULL,

`Дата\_Покупки\_АО` DATE NOT NULL,

`Дата\_Окончания\_Срока\_Эксплуатации` DATE NOT NULL,

Продолжение приложения В

`Количество` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `HWstatus\_Hardware\_idx` (`id\_АО` ASC) VISIBLE,

INDEX `HWstatus\_Company\_idx` (`id\_Компании` ASC) VISIBLE,

CONSTRAINT `HWstatusHardware`

FOREIGN KEY (`id\_АО`)

REFERENCES `KP`.`Hardware` (`id\_АО`)

ON DELETE RESTRICT

ON UPDATE CASCADE,

CONSTRAINT `HWstatusCompany`

FOREIGN KEY (`id\_Компании`)

REFERENCES `KP`.`Company` (`idКомпании`)

ON DELETE RESTRICT

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `KP`.`Orders`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `KP`.`Orders` (

`id\_Заказа` INT NOT NULL AUTO\_INCREMENT,

`id\_Компании` INT NOT NULL,

`id\_АО` INT NULL,

`Количество` INT NULL,

`Одобрено` TINYINT NULL DEFAULT 0,

PRIMARY KEY (`id\_Заказа`),

INDEX `Orders\_Company\_idx` (`id\_Компании` ASC) VISIBLE,

INDEX `Orders\_Hardware\_idx` (`id\_АО` ASC) VISIBLE,

CONSTRAINT `OrdersCompany`

FOREIGN KEY (`id\_Компании`)

REFERENCES `KP`.`Company` (`idКомпании`)

ON DELETE RESTRICT

ON UPDATE CASCADE,

CONSTRAINT `OrdersHardware`

FOREIGN KEY (`id\_АО`)

REFERENCES `KP`.`Hardware` (`id\_АО`)

ON DELETE RESTRICT

Продолжение приложения В

ON UPDATE CASCADE)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

# ПРИЛОЖЕНИЕ Г

(обязательное)

Диаграмма классов пакета Client

|  |
| --- |
|  |
| Рисунок Г.1 Диаграмма классов пакета Client |

Продолжение приложения Г

|  |
| --- |
|  |
| Рисунок Г.2 Диаграмма классов пакета Client |

Продолжение приложения Г

|  |
| --- |
|  |
| Рисунок Г.3 Диаграмма классов пакета Client |

Продолжение приложения Г

|  |  |
| --- | --- |
|  |  |
| Рисунок Г.4 Диаграмма классов пакета Client | |