# Министерство науки и высшего образования РФ

Федеральное государственное образовательное учреждение

высшего образования

«Уфимский государственный авиационный технический университет»

Кафедра автоматизированных систем управления

Направление подготовки

09.03.03 Прикладая информатика

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе по дисциплине «Информационные системы»

«Разработка кроссплатформенного программного продукта на языке JAVA с использованием системы контроля версий»

Выполнил:

Ст. гр. ПИ-223

Батыров Д.Д.

Мингареев Р.А.

Насыров А.Р.

Погудина М.К.

Проверил:

Преподаватель

Казанцев А.В.

Уфа – 2021

# Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

ЗАДАНИЕ

на курсовую работу по дисциплине «Информационные системы»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | Батыров Д.Д. | Группа | ПИ-223 | Консультант | Казанцев А.В. |
|  | Фамилия И.О. |  | номер группы |  | Фамилия И.О. |

|  |  |
| --- | --- |
| 1.Тема курсового  проекта: | Разработка кроссплатформенного программного продукта на языке JAVA |
| с использованием системы контроля версий. |
|  | наименование темы |
| 2.Основное содержание: | |
| 1. Пояснительная записка с необходимыми материалами. 2. Репозиторий системы контроля версий содержащий программный код с комментариями и необходимую документацию. | |

1. Требования к оформлению:
   1. Пояснительная записка должна быть оформлена в текстовом процессоре LibreOffice Writer в соответствии с требованиями СТО УГАТУ. Минимальные требования к оформлению: размер шрифта 14 пунктов; отступы от края листа: отступ слева 2 см. и остальные отступы 0.5 см. В бумажном виде оформляются: титульный лист, задание, календарный план и аннотация, которая содержит ссылку на репозиторий с программным кодом и документацией.
   2. В пояснительной записке должны содержаться следующие разделы:

Раздел 1. Описание предметной области.

Раздел 2. Техническое задание на создание программного продукта.

Раздел 3. Настройка среды разработки для операционных систем семейств Windows и Linux.

Раздел 4. Настройка среды разработки для подключения к системе контроля версий. Раздел 5. Реализация исходного кода по зонам ответственности.

Раздел 6. Сборка и тестирование программного продукта.

Раздел 7. Настройка программной среды для развертывания и запуска программного продукта.

Раздел 8. Руководство пользователя программного продукта.

* 1. В приложение выносится программный код и код тестов.

1. Графическая часть должна включать:

* мнемосхема рассматриваемого процесса;
* диаграммы UML;
* экранные формы инструментальных средств;
* экранные формы, разрабатываемого программного продукта.

Дата выдачи 6 марта 2021 г. Дата окончания 29 мая 2021 г.

Руководитель Казанцев А.В. ФИО

# Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

ЗАДАНИЕ

на курсовую работу по дисциплине «Информационные системы»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | Мингареев Р.А. | Группа | ПИ-223 | Консультант | Казанцев А.В. |
|  | Фамилия И.О. |  | номер группы |  | Фамилия И.О. |

|  |  |
| --- | --- |
| 1.Тема курсового  проекта: | Разработка кроссплатформенного программного продукта на языке JAVA |
| с использованием системы контроля версий. |
|  | наименование темы |
| 2.Основное содержание: | |
| 1. Пояснительная записка с необходимыми материалами. 2. Репозиторий системы контроля версий содержащий программный код с комментариями и необходимую документацию. | |

3.Требования к оформлению:

3.1. Пояснительная записка должна быть оформлена в текстовом процессоре LibreOffice Writer в соответствии с требованиями СТО УГАТУ. Минимальные требования к оформлению: размер шрифта 14 пунктов; отступы от края листа: отступ слева 2 см. и остальные отступы 0.5 см. В бумажном виде оформляются: титульный лист, задание, календарный план и аннотация, которая содержит ссылку на репозиторий с программным кодом и документацией.

* 1. В пояснительной записке должны содержаться следующие разделы:

Раздел 1. Описание предметной области.

Раздел 2. Техническое задание на создание программного продукта.

Раздел 3. Настройка среды разработки для операционных систем семейств Windows и Linux.

Раздел 4. Настройка среды разработки для подключения к системе контроля версий. Раздел 5. Реализация исходного кода по зонам ответственности.

Раздел 6. Сборка и тестирование программного продукта.

Раздел 7. Настройка программной среды для развертывания и запуска программного продукта.

Раздел 8. Руководство пользователя программного продукта.

* 1. В приложение выносится программный код и код тестов.

1. Графическая часть должна включать:

* мнемосхема рассматриваемого процесса;
* диаграммы UML;
* экранные формы инструментальных средств;
* экранные формы, разрабатываемого программного продукта.

Дата выдачи 6 марта 2021 г. Дата окончания 29 мая 2021 г.

Руководитель Казанцев А.В. ФИО

# Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

ЗАДАНИЕ

на курсовую работу по дисциплине «Информационные системы»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | Насыров А.Р. | Группа | ПИ-223 | Консультант | Казанцев А.В. |
|  | Фамилия И.О. |  | номер группы |  | Фамилия И.О. |

|  |  |
| --- | --- |
| 1.Тема курсового  проекта: | Разработка кроссплатформенного программного продукта на языке JAVA |
| с использованием системы контроля версий. |
|  | наименование темы |
| 2.Основное содержание: | |
| 1. Пояснительная записка с необходимыми материалами. 2. Репозиторий системы контроля версий содержащий программный код с комментариями и необходимую документацию. | |

3.Требования к оформлению:

* 1. Пояснительная записка должна быть оформлена в текстовом процессоре LibreOffice Writer в соответствии с требованиями СТО УГАТУ. Минимальные требования к оформлению: размер шрифта 14 пунктов; отступы от края листа: отступ слева 2 см. и остальные отступы 0.5 см. В бумажном виде оформляются: титульный лист, задание, календарный план и аннотация, которая содержит ссылку на репозиторий с программным кодом и документацией.
  2. В пояснительной записке должны содержаться следующие разделы:

Раздел 1. Описание предметной области.

Раздел 2. Техническое задание на создание программного продукта.

Раздел 3. Настройка среды разработки для операционных систем семейств Windows и Linux.

Раздел 4. Настройка среды разработки для подключения к системе контроля версий. Раздел 5. Реализация исходного кода по зонам ответственности.

Раздел 6. Сборка и тестирование программного продукта.

Раздел 7. Настройка программной среды для развертывания и запуска программного продукта.

Раздел 8. Руководство пользователя программного продукта.

* 1. В приложение выносится программный код и код тестов.

1. Графическая часть должна включать:

* мнемосхема рассматриваемого процесса;
* диаграммы UML;
* экранные формы инструментальных средств;
* экранные формы, разрабатываемого программного продукта.

Дата выдачи 6 марта 2021 г. Дата окончания 29 мая 2021 г.

Руководитель Казанцев А.В. ФИО

# Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

ЗАДАНИЕ

на курсовую работу по дисциплине «Информационные системы»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | Погудина М.К. | Группа | ПИ-223 | Консультант | Казанцев А.В. |
|  | Фамилия И.О. |  | номер группы |  | Фамилия И.О. |

|  |  |
| --- | --- |
| 1.Тема курсового  проекта: | Разработка кроссплатформенного программного продукта на языке JAVA |
| с использованием системы контроля версий. |
|  | наименование темы |
| 2.Основное содержание: | |
| 1. Пояснительная записка с необходимыми материалами. 2. Репозиторий системы контроля версий содержащий программный код с комментариями и необходимую документацию. | |

3.Требования к оформлению:

* 1. Пояснительная записка должна быть оформлена в текстовом процессоре LibreOffice Writer в соответствии с требованиями СТО УГАТУ. Минимальные требования к оформлению: размер шрифта 14 пунктов; отступы от края листа: отступ слева 2 см. и остальные отступы 0.5 см. В бумажном виде оформляются: титульный лист, задание, календарный план и аннотация, которая содержит ссылку на репозиторий с программным кодом и документацией.
  2. В пояснительной записке должны содержаться следующие разделы:

Раздел 1. Описание предметной области.

Раздел 2. Техническое задание на создание программного продукта.

Раздел 3. Настройка среды разработки для операционных систем семейств Windows и Linux.

Раздел 4. Настройка среды разработки для подключения к системе контроля версий. Раздел 5. Реализация исходного кода по зонам ответственности.

Раздел 6. Сборка и тестирование программного продукта.

Раздел 7. Настройка программной среды для развертывания и запуска программного продукта.

Раздел 8. Руководство пользователя программного продукта.

* 1. В приложение выносится программный код и код тестов.

1. Графическая часть должна включать:

* мнемосхема рассматриваемого процесса;
* диаграммы UML;
* экранные формы инструментальных средств;
* экранные формы, разрабатываемого программного продукта.

Дата выдачи 6 марта 2021 г. Дата окончания 29 мая 2021 г.

Руководитель Казанцев А.В. ФИО

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Раздел 1. Описание предметной области…………………………………. | 5 |
| Раздел 2. Техническое задание на создание программного продукта….. | 6 |
| [Раздел 3. Настройка среды разработки для операционных систем семейств](#_bookmark3) [Windows и Linux](#_bookmark3)……………………………………………... | 16 |
| [Раздел 4. Настройка среды разработки для подключения к системе контроля](#_bookmark4) [версий](#_bookmark4)………………………………………………………… | 28 |
| [Раздел 5. Реализация исходного кода по зонам ответственности](#_bookmark5)………. | 32 |
| Раздел 6. Сборка и тестирование программного продукта…………….. | 33 |
| Раздел 7. Настройка программной среды для развертывания и запуска программного продукта……………………………………………………. | 36 |
| Раздел 8. Руководство пользователя программного продукта………….. | 38 |
| ПРИЛОЖЕНИЕ 1…………………………………………………………... | 53 |
| ПРИЛОЖЕНИЕ 2………………………………………………………….. | 54 |
| ПРИЛОЖЕНИЕ 3…………………………………………………………... | 84 |
| ПРИЛОЖЕНИЕ 4…………………………………………………………... | 85 |
| ПРИЛОЖЕНИЕ 5…………………………………………………………... | 86 |
| ПРИЛОЖЕНИЕ 6…………………………………………………………... | 87 |
| ПРИЛОЖЕНИЕ 7…………………………………………………………... | 88 |
| СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ……………………….. | 96 |

**Раздел 1. Описание предметной области.**

Тематикой данной курсовой работы является – разработка калькулятора стоимости производства оконных конструкций, который включает в себя такое понятие, как окна в промышленных зданиях.

Данное приложение разрабатывается для компаний по производству окон, конкретнее для менеджеров по продажам и расчету стоимости конечного продукта. В курсовой работе рассматриваются оконные конструкции из следующих материалов: дерево, металл, ПВХ. У пользователей калькулятором есть возможность выбирать размеры окна и дополнительные аксессуары в виде откосов и подоконников. Возможность выбрать механизм открывания створки окна также предусмотрена.

Окно (оконный проём) или витраж — специально задуманная в конструкции здания архитектурная деталь строительства: проём в стене, служащий для поступления света в помещение и/или вентиляции.

Оконные конструкции должны обеспечивать необходимое количество естественного света, требуемое для нормальной работы в помещении. Теплоизоляция и воздухообмен конструкции – важные технические характеристики, на которые стоит обратить внимание при установке окон в промышленное здание.

Планировка производственных зданий и сооружений осуществляется в зависимости от того, какую задачу должен выполняет объект недвижимости. В здания, оборудованные для производства, устанавливаются окна, габариты которых должны соответствовать нормам ГОСТ. Размеры изделий зависят от типа производственного помещения и производимой продукции.

Процесс изготовления начинается с поступления заказа от клиента, в   
качестве которого могут выступать физические и юридические лица. Затем рабочая группа берет замеры оконных проемов. После чего этот заказ обрабатывается инженером-проектировщиком, который работает с заказчиком, учитывает размеры и все требования клиента. С учетом всего этого, а также данных по стандартам и размерам изделия создается чертеж. Оконные конструкции производятся на заводе, поэтому для того чтобы заказ был выполнен, необходима договоренность с поставщиками на поставку сырья на производство, где оно сортируется по материалу профилей, рам и импоста (металл, дерево, пластик). Затем сырье подлежит определенной обработке. После обработки из сырья получаются детали для изготовления изделий. После того как готовы все части профиля, их собирают и получается рама. После того как всё готово к сборке, готовые рамы соединяются импостом, таким образом можно поучить одностворчатые, двухстворчатые или трехстворчатые оконные конструкции. Проверка качества касается как деталей, изделий, так и готовой продукции. Обычный заказ выполняется, в среднем, за 2 недели. Мнемосхема данного бизнес-процесса показана на рисунке 1.



Рисунок 1. Мнемосхема бизнес процесса «Производства оконных конструкций»

На Рисунке 2 представлена диаграмма вариантов использования, описывающая систему на концептуальном уровне.

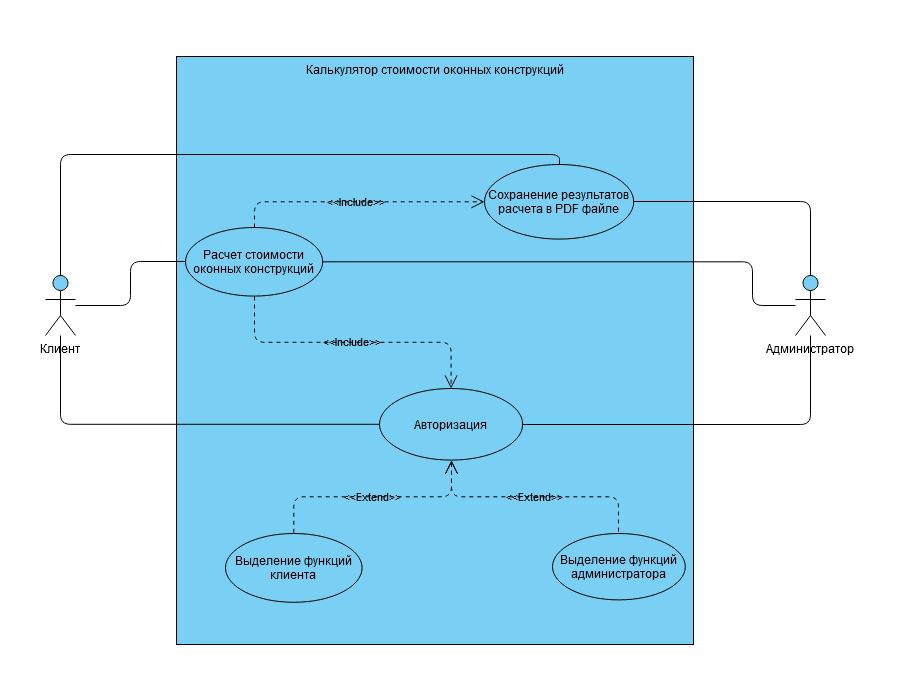


Рисунок 1. Диаграмма вариантов использования

Цены на все материалы представлены в таблице 1.

Таблица 1- Таблица цен

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вид окна | Цена, руб | Тип окна | Цена,  руб | Материалы  рамы | Цена,  руб |
|  |  | Одностворчатое | 3500 | Пластик | 150 |
|  |  | Двухстворчатое | 5500 | Дерево | 300 |
|  |  | Трехстворчатое | 7500 | Металл | 500 |

В соответствии с предметной областью система строится с учетом следующих особенностей:

* изготовление каждого изделия состоит из нескольких стадий;
* стадии заключаются в изготовлении деталей и собирании их в готовое изделие;
* приход и расход сырья определяет наименование поставщика и складирование;
* каждый участок состоит из бригад, бригады — соответственно из рабочих.

Основными документами регулирующие производственные процессы и предъявляющие требования к качеству являются:

ГОСТ 23166-99. Блоки оконные

Настоящий стандарт распространяется на оконные и балконные дверные блоки из древесины, пластмасс и металлических сплавов для зданий и сооружений различного назначения.

ГОСТ 19091-2012. Замки, защелки, механизмы цилиндровые. Методы испытаний

ГОСТ 24700-99. Блоки оконные деревянные со стеклопакетами. Технические условия

ГОСТ 24033-80. Окна и балконные двери деревянные. Методы механических испытаний

ГОСТ 24866-2014. Стеклопакеты клееные. Технические условия

ГОСТ 26602.1-99. Блоки оконные и дверные. Метод определения сопротивления теплопередаче

ГОСТ 26602.2-99. Блоки оконные и дверные. Методы определения воздухо- и водопроницаемости

ГОСТ 26602.3-99\*. Блоки оконные и дверные. Метод определения звукоизоляции

ГОСТ 26602.4-2012. Блоки оконные и дверные. Метод определения общего коэффициента пропускания света

ГОСТ 30698-2014. Стекло закаленное. Технические условия

ГОСТ 12506-81. Габаритные размеры окон для зданий производственных предприятий

В - окна, открывающиеся внутрь помещения;

Г - глухие окна;

Н - открывающиеся наружу.

Примеры габаритов окон указаны на рисунках с 2 по 5.

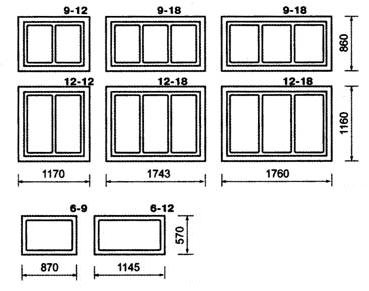


Рисунок 2. Габаритные размеры окон для зданий сельскохозяйственных предприятий

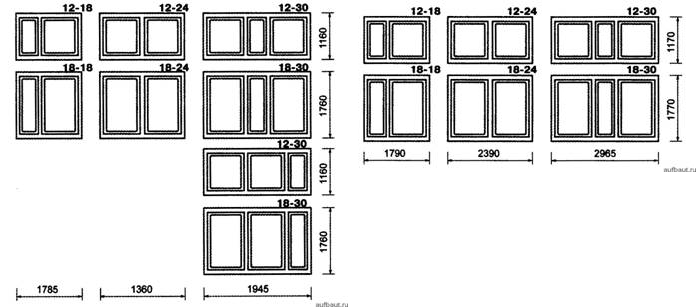


Рисунок 3. Габаритные размеры окон для зданий промышленных предприятий

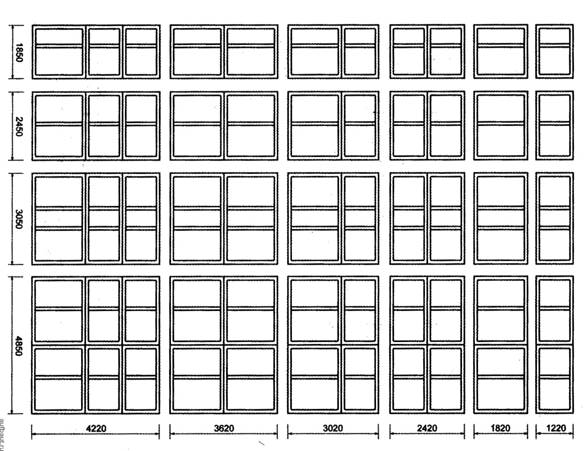


Рисунок 4. Заполнение оконных проемов с ленточным остеклением для промышленных зданий

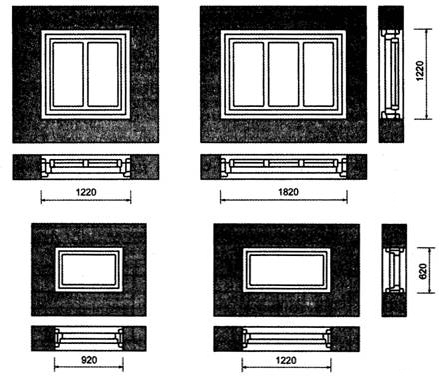


Рисунок 5. Заполнение оконных проемов с простеночным остеклением для зданий сельскохозяйственных предприятий

ФЗ РФ от 07.02.1992 N 2300-1 (ред. от 31.07.2020) "О защите прав потребителей"

Настоящий Закон регулирует отношения, возникающие между потребителями и изготовителями, исполнителями, импортерами, продавцами, владельцами агрегаторов информации о товарах (услугах) при продаже товаров (выполнении работ, оказании услуг), устанавливает права потребителей на приобретение товаров (работ, услуг) надлежащего качества и безопасных для жизни, здоровья, имущества потребителей и окружающей среды, получение информации о товарах (работах, услугах) и об их изготовителях (исполнителях, продавцах), о владельцах агрегаторов информации о товарах (услугах), просвещение, государственную и общественную защиту их интересов, а также определяет механизм реализации этих прав.

Потребитель – гражданин, имеющий намерение заказать или приобрести либо заказывающий, приобретающий или использующий товары (работы, услуги) исключительно для личных, семейных, домашних и иных нужд, не связанных с осуществлением предпринимательской деятельности;

Изготовитель – организация независимо от ее организационно-правовой формы, а также индивидуальный предприниматель, производящие товары для реализации потребителям;

Исполнитель – организация независимо от ее организационно-правовой формы, а также индивидуальный предприниматель, выполняющие работы или оказывающие услуги потребителям по возмездному договору;

Продавец – организация независимо от ее организационно-правовой формы, а также индивидуальный предприниматель, реализующие товары потребителям по договору купли-продажи;

В соответствии со статьей 200 УК РФ, за нарушение Закона «О защите прав потребителей» предусмотрены штрафы в размере от ста до двухсот минимальных окладов, исправительные работы на срок 1-2 года с лишением права заниматься определенной деятельностью или занимать определенные должности на срок до трех лет.

Математический модуль работы программы «Калькулятор стоимости производства оконных конструкций».

Стоимость оконной конструкции вычисляется по формуле 1.

(1)

где, – высота (min= 500мм; max= 3000мм)  
 – ширина (min= 500мм; max= 10000мм)  
 – стоимость рамки

– стоимость стекла в оконном конструкции

**Раздел 3. Настройка среды разработки для операционных систем семейств Windows и Linux**

В данном разделе описаны инструкции по установке и запуску трёх дистрибутивов: Windows 10, Ubuntu 20.04 и OpenSUSE Leap 15.2.

**Настройка операционной системы Windows 10**

Для разработки на данной ОС понадобятся:

1. Eclipse IDE (Integrated Development Environment) – свободная среда разработки модульных кроссплатформенных приложений.

2. JDK (Java Development Kit) – комплект разработчика на языке Java.

3. JRE (Java Runtime Environment) – реализация виртуальной машины, для исполнения Java-приложений, без компилятора и других средств разработки.

Требуется скачать с официального сайта JDK (который содержит JRE) (рис.6):

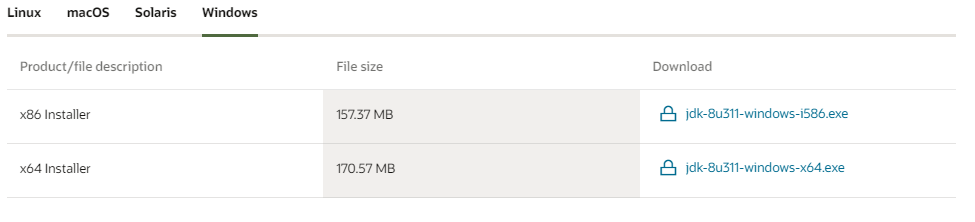


Рисунок 6- Версии JDK.

Далее необходимо установить JDK, следуя пунктам меню, как показано на рисунках 7-10.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7- Установка JDK.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8- Установка JDK.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9- Установка JDK.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10- Установка JDK.

Затем необходимо скачать “Eclipse IDE for Enterprise Java and Web Developers”, версию для ОС Windows, с официального сайта (рис. 11).: <https://www.eclipse.org/downloads/packages/>

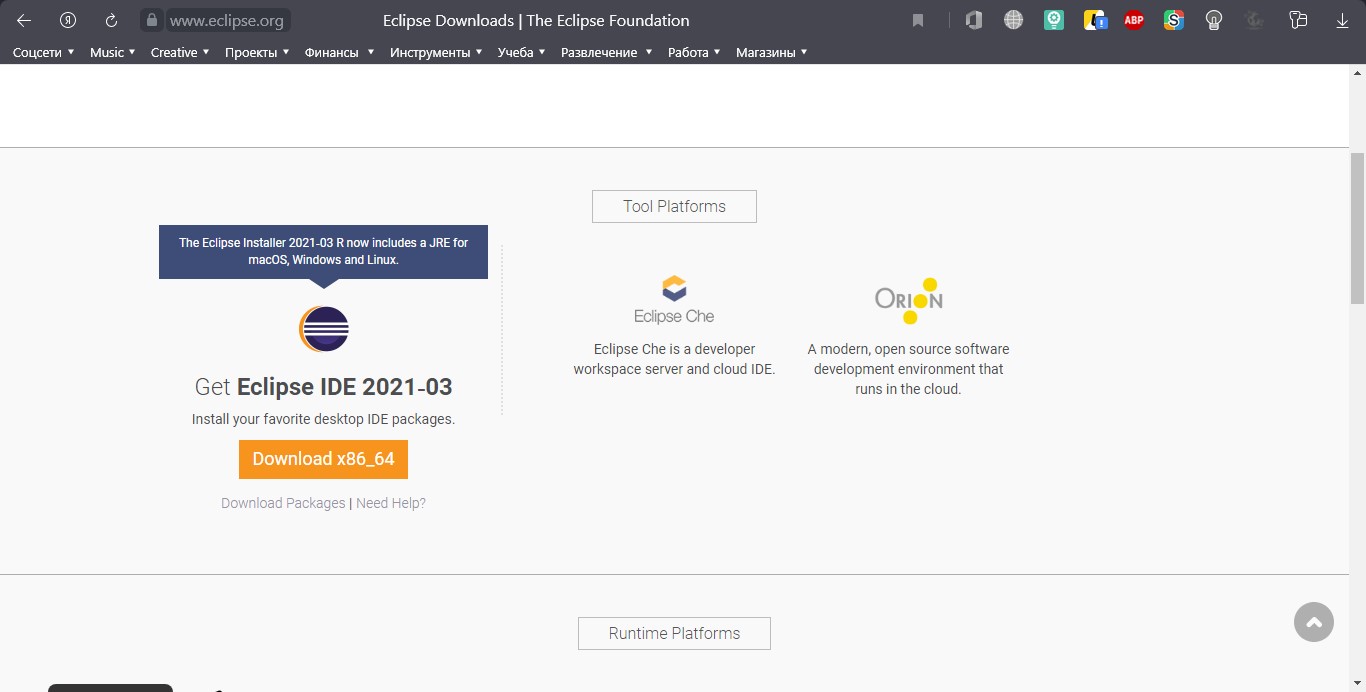


Рисунок 11- Сайт с средой разработки.

Далее необходимо выбрать файл «Eclipse IDE for Java Developers», как продемонстрировано на рисунке 7. После завершения установки, необходимо запустить программу и указать рабочую область.

Чтобы установить Git в Eclipse IDE перейти во вкладку Window, выбрать Perspective – > Open Perspective, как представлено на рисунке 12.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12. Git плагин.

В открывшемся окне выбрать Git и соответствующая проекция появится справа в верхнем углу, перейти в неё. В данной проекции необходимо нажать на «Clone a Git repository», и заполнить следующие поля, как на рисунке 13, где:

URI – ссылка на репозиторий;

User – логин пользователя GitHub.com;

Password – пароль от логина;

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13. Клонирование репозитория через Git.

Затем нажать «Next».

Eclipse IDE скопирует все имеющиеся все данные с репозитория на github.com в локальный репозиторий, что представлено на рисунке 14.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14. Рабочая копия созданная через Git.

Для того чтобы пользоваться функциямиMaven установить maven в свою систему и установить переменные среды Maven.

M2\_HOME:....\apache-maven-3.0.5\maven Установленный путь

M2\_Repo: D:\maven\_repo\Если изменить местоположение репозитория maven

M2:% M2\_HOME%\bin

Шаги по настройке maven на Eclipse IDE:

Открыть Eclipse

Перейти к справочной системе → Eclipse Marketplace (см. рис. 15)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 15. Переход к справочной системе.

Поиск по Maven

Нажать кнопку "Установить" в разделе "Maven Integration for Eclipse", что представлено на рисунке 16.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16. Установка Maven.

Следовать инструкциям при установке.

После успешной установки выполнить следующие действия в Eclipse:

Перейти во вкладку Window → Perspective → Open Perspective. Выбрать Maven для последующей работы с ним.

**Настройка среды разработки Ubuntu 20.04.**

**Установка JDK (JRE).**

Для работы в ОС Ubuntu 20.04 с инструментальной средой Eclipse необходимо выполнить обновление, которое выполняется команда sudo apt upgrade как показано на рисунке 17.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17. Обновление системы

Далее производится установка OpenJDK 8, с помощью команды sudo

apt install openjdk-8-jdk (см. рис. 18)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 18. Установка JDK (JRE).

**Настройка операционной системы Ubuntu 20.04**

Скачать Eclipse IDE с официального сайта, запустить установщик, и после завершения установки указать рабочую область, как на рисунке 19.

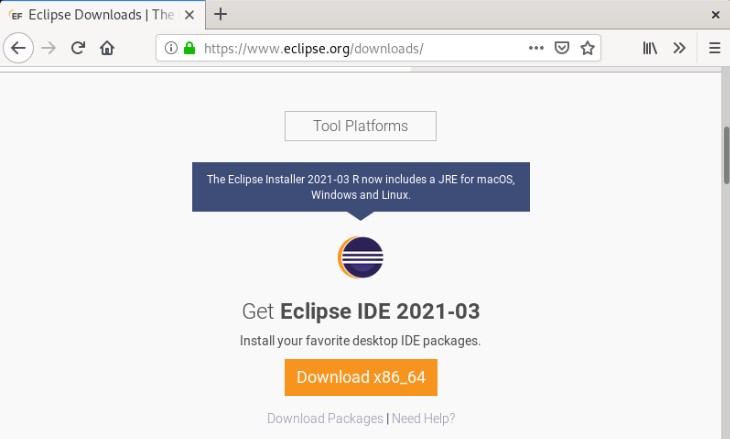


Рисунок 19. Официальный сайт Eclipse открытый через Ubuntu.

Далее необходимо следовать шагам установки.

**3.2.3. Установка Git**

Устанавливаем и настраиваем Git с помощью команд (см. рис. 20):

* sudo apt-get install git
* brew install git

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 20. Установка Git.

Установка Git в Eclipse IDE, подключение к репозиторию на Github и выгрузка с него осуществляется аналогично версии для ОС Windows.

Для создания Maven проекта необходимо выбрать File → New Project → Maven Project, как представлено на рисунке 21.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 21. Окно «New Project»

**Настройка среды разработки OpenSUSE Leap 15.2.**

**Установка JDK (JRE).**

В ОС OpenSUSE Leap 15.2 выполнение обновления выполняется командой sudo zypper update, что показано в соответствии с рисунком 22.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 22. Обновление системы.

После того как будут установлены все требующиеся обновления, нужно ввести команду: sudo zypper install java-1\_8\_0-openjdk (см. рис. 23)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 23. Установка JDK.

**Установка среды разработки Eclipse IDE.**

Скачать Eclipse IDE с официального сайта (см. рис 24), запустить установщик, и после завершения установки указать рабочую область.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 24. Официальный сайт Eclipse.

Далее необходимо следовать шагам установки.

Установка Git в Eclipse IDE, подключение к репозиторию на Github и выгрузка с него осуществляется аналогично версии для ОС Ubuntu 20.04

Установка Maven в Eclipse осуществляется аналогично версии для ОС Ubuntu 20.04.

**Раздел 4. Настройка среды разработки для подключения к системе контроля версий.**

Настройка Git внутри Eclipse IDE на всех операционных системах идентична.

Для использования Git необходимо нажать на кнопку, находящуюся в правом верхнем углу «Open Perspective». Выбрать «Git» (см. рис 25).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 25. «Open Perspective».

В левом окошке откроется выбор из трех вариантов, нажать на Clone a Git repository (см. рисунок 26).

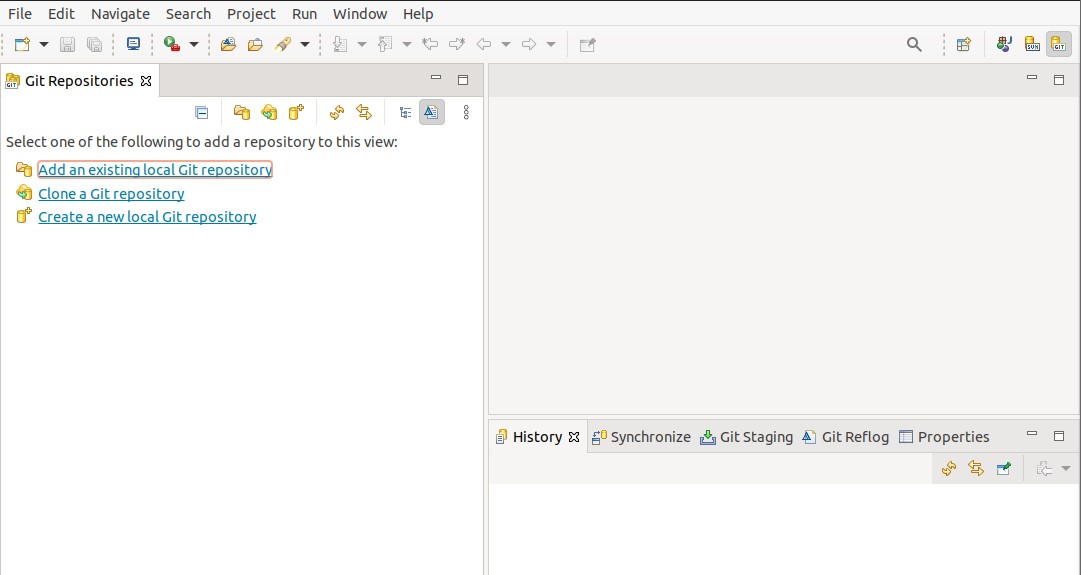


Рисунок 26. Экран добавления нового репозитория.

В открывшимся окне выбрать Clone URL (см. рисунок 6):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 27. Выбор репозитория.

Далее необходимо указать в открывшемся окне:

* URI: https://github.com/BatyrovDD/Project23)
* Host: github.com и путь к репозиторию: /BatyrovDD/project23 Затем ввести логин и пароль от аккаунта Github для авторизации (см. рисунок 28)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 28. Настройка пути репозитория.

На следующем шаге необходимо выбрать нужную ветку и в итоге будет получен локальный репозиторий (см. рисунок 29):

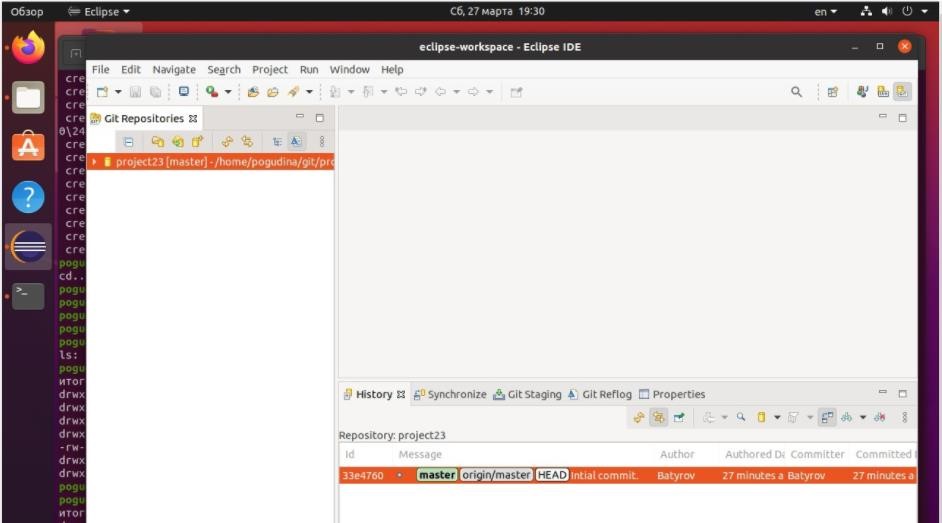


Рисунок 29. Локальный репозиторий.

Для импорта проекта необходимо перейти к «Working Tree», нажать ПКМ и выбрать «Import projects…», где указать путь к каталогу в котором будут располагаться файлы из репозитория.

Для работы с рабочей копией вызывается контекстное меню проекта и далее выбирается вкладка Team, в этой вкладке располагаются все доступные функции (рисунок 30).

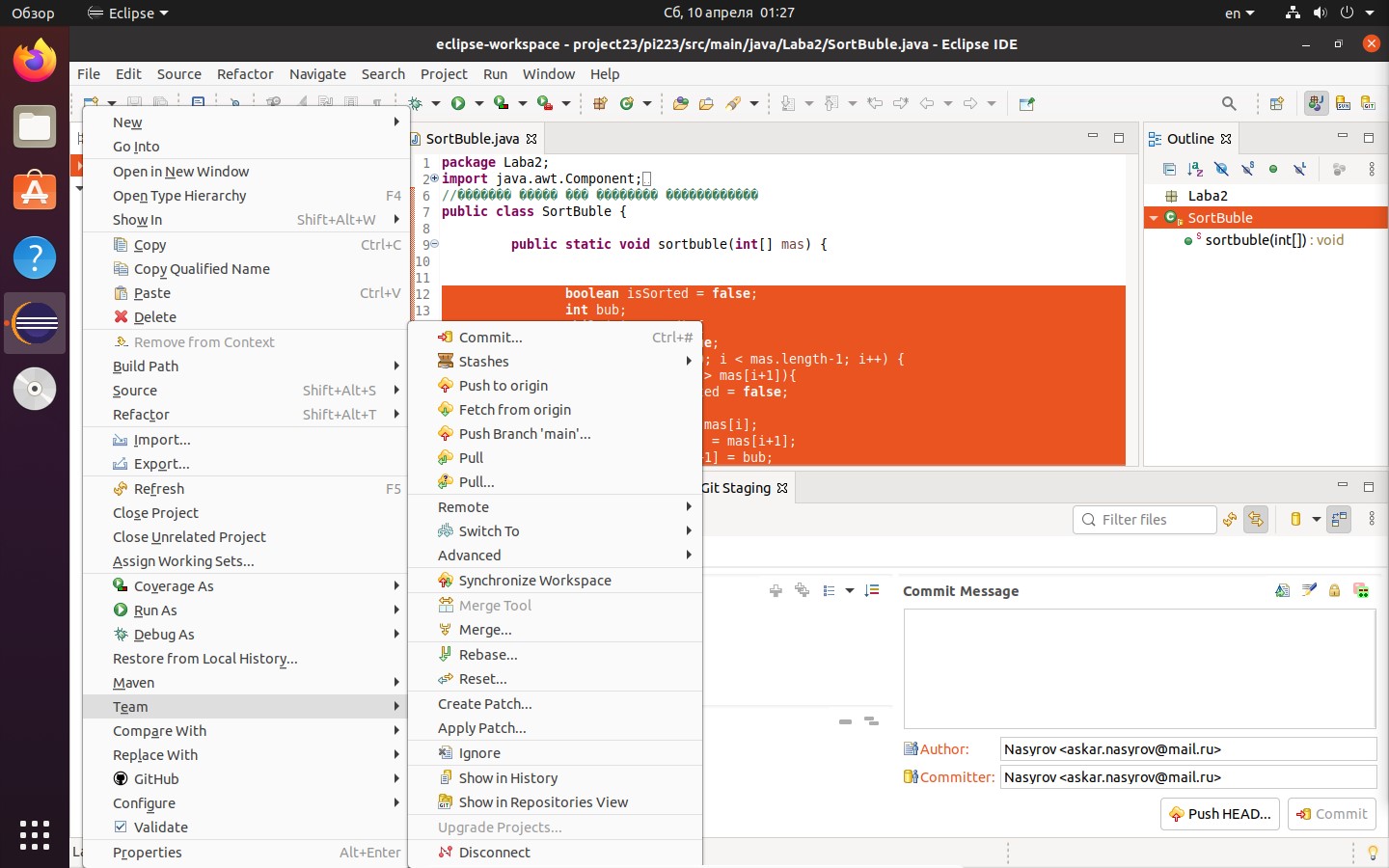


Рис. 30. Вкладка Team.

**Раздел 5. Реализация исходного кода по зонам ответственности.**

Работа по созданию Web-приложения велась по зонам ответственности, представленным в таблице:

Таблица 3 — Зоны ответственности

|  |  |  |
| --- | --- | --- |
| **№** | **ФИО разработчика/ модератора** | **Зона ответственности** |
| 1 | Батыров Денис Дамирович | Разработка JavaScript-кода для реализации динамической работы веб-приложения |
| 2 | Мингареев Радмир Адикович | Разработка серверного приложения и интерфейса API |
| 3 | Насыров Аскар  Русланович | Разработка страниц веб-приложения (логин, сохранение паролей, выход, регистрация |
| 4 | Погудина Милена Константиновна | Разработка PDF для формирования отчета в PDF формате, Разработка DAO базы данных и механизма хранения. |

**Раздел 6. Сборка и тестирование программного продукта**

Описание UNIT-тестов представлено в таблице 4.

Таблица 4 – Описание UNIT-тестов

|  |  |  |  |
| --- | --- | --- | --- |
| № | ФИО разработчика/  модератор | Описание UNIT-теста | Номер приложения |
| 1 | Батыров Денис Дамирович | Тест проверяет, что при неправильном формировании ответа в формате JSON в его заголовке обязательно устанавливается атрибут со значением «ложь». | см. прило-  жение П3 |
| 2 | Мингареев Радмир Адикович | Тест проверяет, что при каждом вызове метода генерации случайных строк форми-руется новая строка определенной длины. | см. прило-  жение П4 |
| 3 | Насыров Аскар Русланович | Тест проверяет, что все установщики и получатели атрибутов класса «User» рабо-тают корректно. | см. прило-  жение П5 |
| 4 | Погудина Милена Константиновна | Тест проверяет, что все установщики и получатели атрибутов класса «Password» рабо-тают корректно. | см. прило-  жение П6 |

Описание структуры проекта по каталогам, как показано на рисунке 17, проект включает в себя такие каталоги как:

− artifacts – содержит артефакты (результат разработки). В данном проекте артефактом является файл Calculate.war. В данном каталоге так же содержится контейнер сервлетов webapp-runner.java;

− src/main/java – содержит исходный программный код, разложенный по пакетам в виде файлов с расширением java (WriterInFile.java, ArcHangar.java, AuthBaseController.java, AuthManager.java, Authorization.java, BoxHangar.java, Calculator.java, Hangar.java, PDFWriter.java, TentHangar.java);

− src/main/webapp – содержит компоненты, относящиеся к веб-части;

− src/test – содержит unit-тесты;

− target – содержит выходную информацию при сборке проекта.

Такой каталог как src/main/webapp содержит следующие каталоги, которые перечислены ниже:

− Каталог css, содержащий css стиль под названием style.css, определяющий стиль веб-страниц.

− Каталог WEB-INF. В нём расположен файл web.xml, определяющий конфигурацию веб-приложения при развёртывании.

− Jsp-страницы Authorization.jsp, Edit.jsp, Form.jsp, Results.jsp. Эти файлы определяют содержание веб-страниц.

Так же данный проект включает в себя файлы, которые предоставлены ниже:

− README.md – описание проекта.

− pom.xml – конфигурация сборки Maven проекта.

− PROCFILE – конфигурация для сервиса Heroku.

− src/main/AuthBase.txt – файл с первоначальной базой учётных записей пользователей.

В файле pom.xml описывается конфигурация для сборки проекта Maven'ом. В описание включается:

− Описание проекта (его название, версия и т.д.).

− После идёт описание всех плагинов: плагин компиляции (maven-compiler-plugin), плагины maven (maven-compiler-plugin) с изменением конфигурации в виде добавления артефакта com.heroku.webapp-runner, плагин для компиляции war-файлов (maven-war-plugin).

− Так же включено описание зависимостей: модульное тестирование (junit), сервлеты (javax.servlet), Unified Expression Language (el-api), создание pdf файлов (com.itextpdf), контейнер сервлетов (com.github.jsimone.webapp-runner).

**Процесс сборки проекта:**

Сборка проекта будет проходить через сборщик проектов Maven. Для начала нужно запустить Eclipse IDE и скачать проект с репозитория. Необходимо нажать на весь проект, найти вкладку «Run as» и выбрать «Maven build…», как показано на рисунке 31.

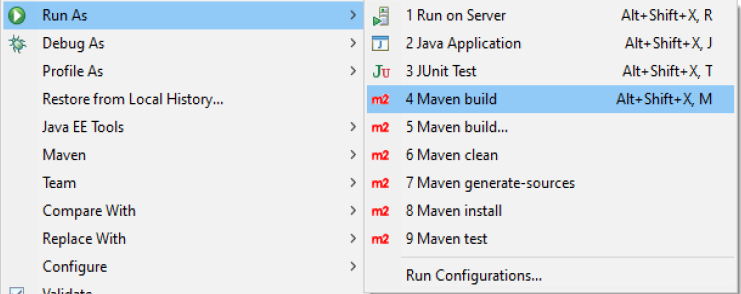


Рисунок 32 – Работа в Eclipse

В открывшемся окне ввести в поле Goals «package» и нажать «Run».

В результате получаем файл, который будет хранится в artifacts.

**Раздел 7. Настройка программной среды для развертывания и запуска программного продукта**

Для реализации системы непрерывной интеграции и доставки была выбрана программная система *Jenkins*, работающая на *Java* и имеющая открытый исходный код.

На сервер была установлена *Jenkins* версии 2.277.4, а также необходимые файлы для сборки и тестирования *Java*-приложений: *JDK* версии 8 и сборщик проектов *Maven* 3.6.3 (Рисунок 33).

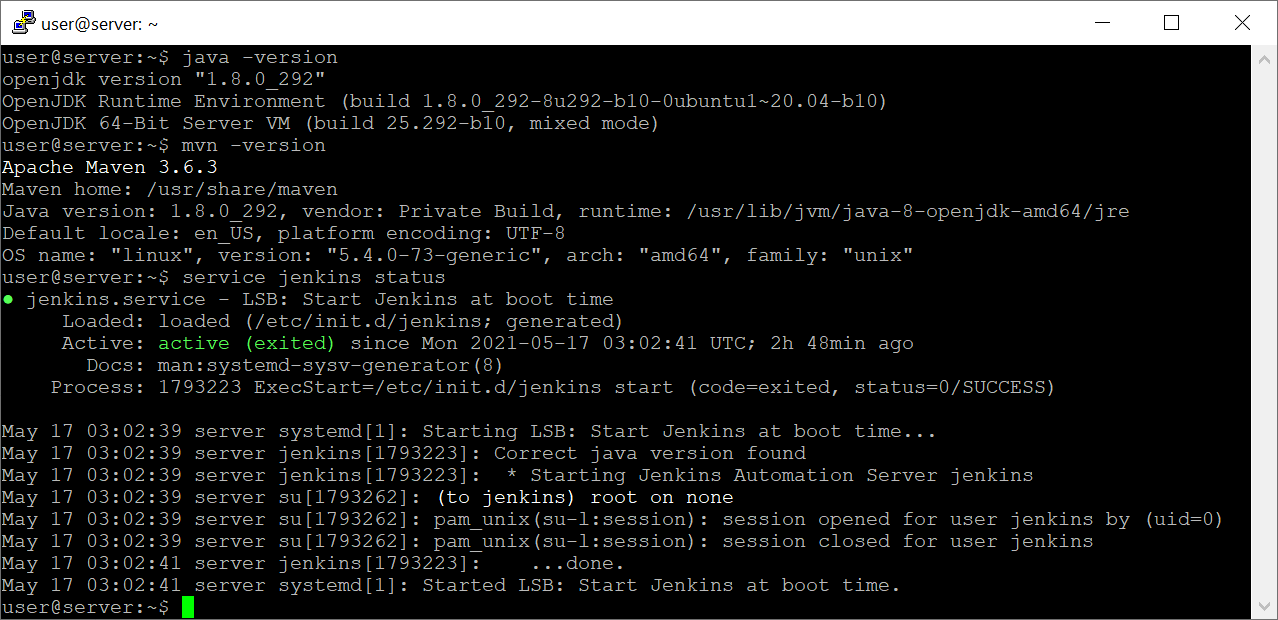


Рисунок 33. Установленное ПО на сервере

Далее через веб-интерфейс была произведена начальная настройка *Jenkins* и установлены плагины. Была настроена маршрутизация для внешнего доступа к Jenkins по адресу: [*https://project23.usatu.su/jenkins/*](https://project23.usatu.su/jenkins/)

В *Jenkins* был создан проект под названием «Project23\_CID». В его настройках указывается конфигурация шагов сборки. В них записывались адрес репозитория, идентификационные данные, название ветки, изменения в которой начнут процесс интеграции и доставки

В качестве *Build Trigger* (тип события, начинающее процесс интеграции) был выбран «GitHub hook trigger for GITScm polling» (Рисунок 34).

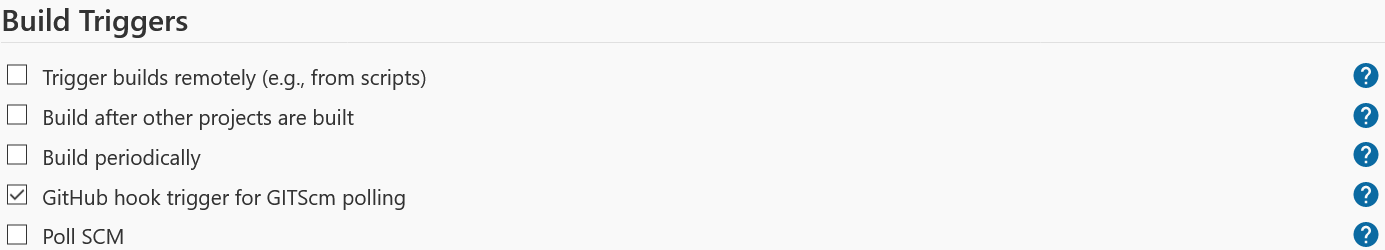


Рисунок 34. Выбор типа события, инициализирующее начало интеграции

В разделе *Build* была указана конфигурация для сборки и тестирования (Рисунок 35).



Рисунок 35. Конфигурация сборки и тестирования

А также конфигурация для развертывания собранного *war*-архива и других файлов приложения (в виде *.sh* скрипта)

В настройках репозитория GitHub был настроено событие отправки запроса, происходящее при фиксации изменений (webhook)

На этом настройка непрерывной интеграции и доставки была завершена. Теперь после каждой фиксации в репозитории *Jenkins* автоматически собирает *war*-архив разрабатываемого приложения и выгружает его и остальные файлы на сервер.

Документ «Руководство оператора» представлен в Приложении 6.

**ПРИЛОЖЕНИЕ 1**

ФГБОУ ВО УГАТУ УФИМСКИЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ ГОСУДРАСТВЕННЫЙ УНИВЕРСИТЕТ

СОГЛАСОВАНО УТВЕРЖДАЮ

Доцент кафедры АСУ ФГБОУ Студент группы ПИ-223 ФИРТ ВО УГАТУ ФГБОУ ВО УГАТУ, модератор

Казанцев А. В. Батыров Д. Д.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Личная | Расшифровка | Личная |  | Расшифровка |
| подпись | подписи | подпись |  | подписи |
|  | 27.03.2021 |  | 27.03.2021 |  |

**КАЛЬКУЛЯТОР РАСЧЁТА СТОИМОСТИ ОКОННЫХ КОНСТРУКЦИЙ**

|  |  |
| --- | --- |
| *Подпись и дата* |  |
| *Инв. № дубл.* |  |
| *Взам. инв. №* |  |
| *Подпись и дата* |  |
| *Инв. № подл.* |  |

**Техническое задание**

**ЛИСТ УТВЕРЖДЕНИЯ**

**643.02069438.00001-01 ТЗ 01-ЛУ**

**(Электронный)**

СОГЛАСОВАНО Представитель команды разработчиков Доцент кафедры АСУ ФГБОУ Студент группы ПИ-223 ФИРТ

ВО УГАТУ ФГБОУ ВО УГАТУ, модератор

Казанцев А. В. Батыров Д. Д.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Личная | Расшифровка | Личная |  | Расшифровка |
| подпись | подписи | подпись |  | подписи |
|  | 27.03.2021 |  | 27.03.2021 |  |

УТВЕРЖДЕН 643.02069438.00001-01 ТЗ 01-ЛУ

**КАЛЬКУЛЯТОР РАСЧЁТА СТОИМОСТИ ОКОННЫХ КОНСТРУКЦИЙ**

|  |  |
| --- | --- |
| *Подпись и дата* |  |
| *Инв. № дубл.* |  |
| *Взам. инв. №* |  |
| *Подпись и дата* |  |
| *Инв. № подл.* |  |

**Техническое задание 643.02069438.00001-01 ТЗ 01**

**Листов 22**

**СОДЕРЖАНИЕ**

1. [ВВЕДЕНИЕ 4](#_bookmark10)
2. [ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ](#_bookmark13) 5
3. [НАЗНАЧЕНИЕ РАЗРАБОТКИ 6](#_bookmark16)
   1. [Функциональное назначение программы](#_bookmark17) 6
   2. [Эксплуатационное назначение программы](#_bookmark18) 6
4. [ТРЕБОВАНИЯ К ПРОГРАММЕ](#_bookmark19) 7
   1. [Требования к функциональным характеристикам](#_bookmark20) 7
      1. [Требования к составу выполняемых функций](#_bookmark21) 7
      2. [Требования к организации входных данных](#_bookmark22) 7
      3. [Требования к организации выходных данных](#_bookmark23) 7
      4. [Требования к временным характеристикам](#_bookmark24) 7
   2. [Требования к надежности](#_bookmark25) 7
      1. [Требования к обеспечению устойчивого функционирования программы](#_bookmark26) 7
      2. [Требования к защите информации от несанкционированного доступа](#_bookmark29) 8
   3. [Условия эксплуатации](#_bookmark30) 8
      1. [Климатические условия эксплуатации](#_bookmark31) 8
      2. [Требования к видам обслуживания](#_bookmark32) 8
      3. [Требования к численности и квалификации персонала](#_bookmark33) 8
   4. [Требования к составу и параметрам технических средств](#_bookmark34) 9
   5. [Требования к информационной и программной совместимости 10](#_bookmark35)
      1. [Требования к информационным структурам и методам решения 1](#_bookmark36)0
      2. [Требования к исходным кодам и языкам программирования 1](#_bookmark37)0
      3. [Требования к программным средствам, используемым программой 1](#_bookmark38)0
   6. [Требования к маркировке и упаковке 1](#_bookmark39)0
   7. [Требования к транспортированию и хранению 1](#_bookmark40)0
   8. [Специальные требования 1](#_bookmark41)0
5. [ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕТАЦИИ 1](#_bookmark42)1
6. [ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 1](#_bookmark44)2
   1. [Ориентированная экономическая эффективность 1](#_bookmark45)2
   2. [Предполагаемая годовая потребность 1](#_bookmark46)2
   3. [Экономические преимущества разработки 1](#_bookmark47)2
7. [СТАДИИ И ЭТАПЫ РАЗРАБОТКИ 1](#_bookmark48)3
8. [ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ](#_bookmark52) 14

# ВВЕДЕНИЕ

Калькулятор «Расчёт стоимости оконных конструкций для промышленных предприятий» – это программа, позволяющая определить предварительную цену оконной конструкции с определенными выбранными параметрами.

Оконная конструкция – это светопропускающая конструкция, состоящая из остеклённых рамных элементов (коробок, створок, фрамуг).

# ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Основанием для разработки данного ПО послужило задание на курсовую работу.

# НАЗНАЧЕНИЕ РАЗРАБОТКИ



# ТРЕБОВАНИЯ К ПРОГРАММЕ



организационно-технических мероприятий, перечень которых приведен ниже:

1) организацией бесперебойного питания технических средств (для хостинга, где расположена программа, либо же для персонального компьютера);

2) необходимым уровнем квалификации пользователей.

**4.2.2. Требования к защите информации от несанкционированного доступа**

Для защиты информации от несанкционированного доступа программа должна обеспечивать идентификацию и аутентификацию пользователя.

**4.3. Условия эксплуатации**

**4.3.1. Климатические условия эксплуатации**

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

**4.3.2. Требования к видам обслуживания**

См. Требования к обеспечению надежного (устойчивого) функционирования программы.

**4.3.3. Требования к численности и квалификации персонала**

Количество персонала, требуемого для работы программы, должно составлять не менее 1 человека – администратора или конечного пользователя программы – оператора.

Администратор должен иметь высшее профильное образование в сфере IT. Пользователь должен иметь навыки использования графического интерфейса ОС.

# Требования к составу и параметрам технических средств

Конфигурация сервера «https://project23.usatu.su/» представлена в соответствии с рисунком 1.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 1 – Конфигурация сервера «https://project23.usatu.su/»



# ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕТАЦИИ



# ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

# Ориентированная экономическая эффективность

Ориентировочная экономическая эффективность не рассчитывается.

# Предполагаемая годовая потребность

Предполагаемое число использования программы в год – около 150 раза в год.

# Экономические преимущества разработки

Экономические преимущества разработки не рассчитываются.

# СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

# Стадии и этапы разработки

Стадии и этапы разработки представлены в таблице 3.

Таблица 3 – Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа работ | Трудоемкость выполнения, час | Процент к общей трудоемкости  выполнения | Срок предъявления консультанту |
| Получение и согласование задания | 1,7 | 1,7% | 27 неделя |
| Раздел 1. Описание предметной области | 20 | 20% | 29 неделя |
| Раздел 2. Техническое задание на создание  программного продукта | 10 | 10% | 30 неделя |
| Раздел 3. Настройка среды разработки для  операционных систем  семейств Windows и Linux | 10 | 10% | 31 неделя |
| Раздел 4. Настойка среды  разработки для подключения к системе контроля версий | 7 | 7% | 32 неделя |
| Раздел 5. Реализация  исходного кода по зонам ответственности | 23 | 23% | 34 неделя |
| Раздел 6. Сборка и тестирование программного  продукта | 8 | 8% | 35 неделя |
| Раздел 7. Настройка программной среды для развертывания и запуска  программного продукта | 10 | 10% | 36 неделя |
| Раздел 8. Руководство  пользователя программного продукта | 10 | 10% | 37 неделя |
| Защита | 0,3 | 0,3% | 38 неделя |

# ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ



**ПРИЛОЖЕНИЕ 2**

**Controller**

**Default**

package su.usatu.project23.controller;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import su.usatu.project23.util.JsonResponseUtil;

@WebServlet("/")

public class Default extends HttpServlet {

public Default() {

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "API method not found");

response.setStatus(HttpServletResponse.SC\_NOT\_FOUND);

out.println(jsonOutput);

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "API method not found");

response.setStatus(HttpServletResponse.SC\_NOT\_FOUND);

out.println(jsonOutput);

}

}

**Login**

package su.usatu.project23.controller;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import su.usatu.project23.dao.Project23DAO;

import su.usatu.project23.dao.Project23DAOImplementation;

import su.usatu.project23.model.User;

import su.usatu.project23.util.JsonResponseUtil;

@WebServlet("login")

public class Login extends HttpServlet {

private Project23DAO dao;

public Login() {

dao = new Project23DAOImplementation();

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

User user = new User();

String username = request.getParameter("username");

String password = request.getParameter("password");

if (username == "" || password == "") {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Login failed: missing required parameters");

response.setStatus(HttpServletResponse.SC\_UNAUTHORIZED);

} else {

user.setPassword(password);

user.setUsername(username);

boolean passwordIsCorrect = dao.checkLoginPasswordMatch(user, "users");

if (!passwordIsCorrect) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Login failed: wrong username or password");

response.setStatus(HttpServletResponse.SC\_UNAUTHORIZED);

} else {

User loggedUser = new User();

User userInfo = new User();

loggedUser = dao.getUserByUsername(username, "users");

String token = loggedUser.getApiToken();

userInfo = dao.getUserInfoByToken(token, "users");

jsonOutput = JsonResponseUtil.formJsonResponse("success", "Login successful", userInfo);

}

}

out.println(jsonOutput);

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

//GET is not allowed on this page

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Operation failed: only POST allowed");

response.setStatus(HttpServletResponse.SC\_METHOD\_NOT\_ALLOWED);

out.println(jsonOutput);

}

}

**PDFGenerator**

package su.usatu.project23.controller;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import su.usatu.project23.dao.Project23DAO;

import su.usatu.project23.dao.Project23DAOImplementation;

import su.usatu.project23.model.ReportData;

import su.usatu.project23.util.JsonResponseUtil;

@WebServlet("generate\_pdf")

public class PDFGenerator extends HttpServlet {

private Project23DAO dao;

public PDFGenerator() {

dao = new Project23DAOImplementation();

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

ReportData rd = new ReportData();

String pdfReport;

String token = request.getParameter("token");

rd.width = request.getParameter("width");

rd.height = request.getParameter("height");

rd.sashesCount = request.getParameter("sashesCount");

rd.glazing = request.getParameter("glazing");

rd.totalAmount = request.getParameter("totalAmount");

pdfReport = dao.createPdfReport(rd);

if (pdfReport == "Failed") {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Failed to create PDF report");

response.setStatus(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR); // Status code 500

} else {

jsonOutput = JsonResponseUtil.formJsonResponse("success", "OK", pdfReport);

dao.assignPdfReportToUser(token, pdfReport);

}

out.println(jsonOutput);

}

}

**PDFGetter**

package su.usatu.project23.controller;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import su.usatu.project23.dao.Project23DAO;

import su.usatu.project23.dao.Project23DAOImplementation;

import su.usatu.project23.util.JsonResponseUtil;

@WebServlet("get\_user\_reports")

public class PDFGetter extends HttpServlet {

private Project23DAO dao;

public PDFGetter() {

dao = new Project23DAOImplementation();

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

int userId = Integer.parseInt(request.getParameter("ownerId"));

String[] outputArray = dao.getUserPdfFiles(userId);

if (outputArray.length == 0) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Docs not found");

response.setStatus(HttpServletResponse.SC\_NOT\_FOUND); // Status code 404

} else {

jsonOutput = JsonResponseUtil.formJsonResponse("success", "Found", outputArray);

}

out.println(jsonOutput);

}

}

**PDFRemover**

package su.usatu.project23.controller;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import su.usatu.project23.dao.Project23DAO;

import su.usatu.project23.dao.Project23DAOImplementation;

import su.usatu.project23.util.JsonResponseUtil;

@WebServlet("delete\_pdf\_report")

public class PDFRemover extends HttpServlet {

private Project23DAO dao;

public PDFRemover() {

dao = new Project23DAOImplementation();

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

String token = request.getParameter("token");

String documentName = request.getParameter("documentName");

boolean deletePdf = dao.deletePdfReport(token, documentName);

if (!deletePdf) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Document not found");

response.setStatus(HttpServletResponse.SC\_NOT\_FOUND); // Status code 404

} else {

jsonOutput = JsonResponseUtil.formJsonResponse("success", "Document deleted sucessfully", "");

}

out.println(jsonOutput);

}

}

**RatesEditor**

package su.usatu.project23.controller;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import su.usatu.project23.dao.\*;

import su.usatu.project23.model.Rates;

import su.usatu.project23.model.User;

import su.usatu.project23.util.JsonResponseUtil;

@WebServlet("edit\_prices")

public class RatesEditor extends HttpServlet {

private Project23DAO dao;

public RatesEditor() {

dao = new Project23DAOImplementation();

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

String token = request.getParameter("token");

User user = dao.getUserByToken(token, "users");

int groupId = user.getGroupId();

Rates rates = new Rates();

rates.id = Integer.parseInt(request.getParameter("rates\_set\_id"));

rates.single\_glazing\_price = Double.parseDouble(request.getParameter("single\_rate\_price"));

rates.double\_glazing\_price = Double.parseDouble(request.getParameter("daily\_rate\_price"));

rates.triple\_glazing\_price = Double.parseDouble(request.getParameter("night\_rate\_price"));

if (groupId == 1) {

dao.editRates(token, rates);

jsonOutput = JsonResponseUtil.formJsonResponse("success", "Изменения внесены", null);

out.println(jsonOutput);

} else {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Access Denied");

response.setStatus(HttpServletResponse.SC\_FORBIDDEN);

out.println(jsonOutput);

}

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

**RatesGetter**

package su.usatu.project23.controller;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import su.usatu.project23.dao.\*;

import su.usatu.project23.model.Rates;

import su.usatu.project23.util.JsonResponseUtil;

@WebServlet("get\_prices")

public class RatesGetter extends HttpServlet {

private Project23DAO dao;

public RatesGetter() {

dao = new Project23DAOImplementation();

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

int rateSetId;

if (request.getParameter("rates\_set\_id").trim().isEmpty()) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "'rates\_set\_id' parameter is required");

response.setStatus(HttpServletResponse.SC\_BAD\_REQUEST);

} else {

String ratesIdinString = request.getParameter("rates\_set\_id");

rateSetId = Integer.parseInt(ratesIdinString);

if (!(rateSetId >= 1 && rateSetId < 4)) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Inconsistent 'rates\_set\_id' parameter");

response.setStatus(HttpServletResponse.SC\_BAD\_REQUEST);

} else {

Rates selectedRates = dao.getRatesById(rateSetId);

jsonOutput = JsonResponseUtil.formJsonResponse("success", "Found", selectedRates);

}

}

out.println(jsonOutput);

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// POST is not allowed on this page

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Operation failed: only GET allowed");

response.setStatus(HttpServletResponse.SC\_METHOD\_NOT\_ALLOWED);

out.println(jsonOutput);

}

}

**Register**

package su.usatu.project23.controller;

import java.io.IOException;

import java.io.PrintWriter;

import java.security.NoSuchAlgorithmException;

import java.security.spec.InvalidKeySpecException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import su.usatu.project23.dao.Project23DAO;

import su.usatu.project23.dao.Project23DAOImplementation;

import su.usatu.project23.model.User;

import su.usatu.project23.util.\*;

@WebServlet("register")

public class Register extends HttpServlet {

private Project23DAO dao;

public Register() {

dao = new Project23DAOImplementation();

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// GET is not allowed on this page

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Operation failed: only POST allowed");

response.setStatus(HttpServletResponse.SC\_METHOD\_NOT\_ALLOWED);

out.println(jsonOutput);

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

User user = new User();

String salt = PasswordUtil.getSalt();

user.setUsername(request.getParameter("username"));

user.setEmail(request.getParameter("email"));

user.setFullName(request.getParameter("full\_name"));

boolean usernameIsUnique = dao.checkDbValueIfUnique("username", user.getUsername(), "users");

boolean emailIsUnique = dao.checkDbValueIfUnique("email", user.getEmail(), "users");

if (!usernameIsUnique) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "The username is already taken");

response.setStatus(HttpServletResponse.SC\_CONFLICT);

} else if (!emailIsUnique) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "The email address is already taken");

response.setStatus(HttpServletResponse.SC\_CONFLICT);

} else {

try {

String password = request.getParameter("password");

String hashedPassword = PasswordUtil.hashPassword(password, salt);

String apiKey = TokenUtil.generateNewToken();

user.setPassword(hashedPassword);

user.setSalt(salt);

user.setApiToken(apiKey);

boolean userAdded = dao.addUser(user, "users");

if (!userAdded) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "SQLException");

response.setStatus(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);

} else {

User userInfo = new User();

userInfo = dao.getUserInfoByToken(apiKey, "users");

jsonOutput = JsonResponseUtil.formJsonResponse("success", "Registration successful", userInfo);

}

} catch (NoSuchAlgorithmException | InvalidKeySpecException e) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", e.getMessage());

response.setStatus(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);

e.printStackTrace();

}

}

out.println(jsonOutput);

}

}

**UserGetter**

package su.usatu.project23.controller;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import su.usatu.project23.dao.Project23DAO;

import su.usatu.project23.dao.Project23DAOImplementation;

import su.usatu.project23.model.User;

import su.usatu.project23.util.JsonResponseUtil;

@WebServlet("get\_user\_info")

public class UserGetter extends HttpServlet {

private Project23DAO dao;

public UserGetter() {

dao = new Project23DAOImplementation();

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

String apiKey = request.getParameter("token");

User user = dao.getUserByToken(apiKey, "users");

if (user.getId() == 1) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "User Not Found");

response.setStatus(HttpServletResponse.SC\_NOT\_FOUND); // Status code 404

} else {

jsonOutput = JsonResponseUtil.formJsonResponse("success", "User found", user);

}

out.println(jsonOutput);

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

}

}

**UserUpdater**

package su.usatu.project23.controller;

import java.io.IOException;

import java.io.PrintWriter;

import java.security.NoSuchAlgorithmException;

import java.security.spec.InvalidKeySpecException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import su.usatu.project23.dao.Project23DAO;

import su.usatu.project23.dao.Project23DAOImplementation;

import su.usatu.project23.model.User;

import su.usatu.project23.util.JsonResponseUtil;

import su.usatu.project23.util.PasswordUtil;

import su.usatu.project23.util.TokenUtil;

@WebServlet("update\_profile")

public class UserUpdater extends HttpServlet {

private Project23DAO dao;

public UserUpdater() {

dao = new Project23DAOImplementation();

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

String token = request.getParameter("token");

User user = new User();

user = dao.getUserByToken(token, "users");

if (user.getId() == 1) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "User Not Found");

response.setStatus(HttpServletResponse.SC\_NOT\_FOUND); // Status code 404

out.println(jsonOutput);

return;

}

String userEmail = request.getParameter("email");

user.setEmail(userEmail);

String userFullName = request.getParameter("fullName");

user.setFullName(userFullName);

String password = request.getParameter("password");

boolean passwordUpdate = password.equals("111111111") ? false : true;

if (passwordUpdate) {

try {

String salt = PasswordUtil.getSalt();

String hashedPassword = PasswordUtil.hashPassword(password, salt);

String apiKey = TokenUtil.generateNewToken();

user.setPassword(hashedPassword);

user.setSalt(salt);

user.setApiToken(apiKey);

} catch (NoSuchAlgorithmException | InvalidKeySpecException e) {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", e.getMessage());

response.setStatus(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR); // Status code 500

out.println(jsonOutput);

return ;

}

}

if (dao.updateUser(token, user)) {

User updatedUser = new User();

String username = user.getUsername();

updatedUser = dao.getUserByUsername(username, "users");

String newToken = updatedUser.getApiToken();

User userInfo = new User();

userInfo = dao.getUserInfoByToken(newToken, "users");

jsonOutput = JsonResponseUtil.formJsonResponse("success", "Operation successful", userInfo);

out.println(jsonOutput);

} else {

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Update failed");

response.setStatus(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR); // Status code 500

out.println(jsonOutput);

return;

}

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// GET is not allowed on this page

response.setContentType("application/json");

PrintWriter out = response.getWriter();

String jsonOutput;

jsonOutput = JsonResponseUtil.formJsonResponse("failure", "Operation failed: only POST allowed");

response.setStatus(HttpServletResponse.SC\_METHOD\_NOT\_ALLOWED);

out.println(jsonOutput);

}

}

**Project23DAO**

package su.usatu.project23.dao;

import java.io.IOException;

import su.usatu.project23.model.\*;

public interface Project23DAO {

public Rates getRatesById(int id);

public boolean editRates(String token, Rates rates);

public boolean addUser(User user, String tableName);

public User getUserByToken(String token, String tableName);

public User getUserByUsername(String username, String tableName);

public User getUserInfoByToken(String token, String tableName);

public boolean updateUser(String token, User user);

public boolean checkDbValueIfUnique(String rowLabel, String value, String tableName);

public boolean checkLoginPasswordMatch(User user, String tableName);

public String createPdfReport(ReportData dataForPDF) throws IllegalStateException, IOException;

public boolean assignPdfReportToUser(String token, String documentName);

public String[] getUserPdfFiles(int userId);

public boolean deletePdfReport(String token, String documentName);

}

**Project23DAOImplementation**

package su.usatu.project23.dao;

import java.io.IOException;

import java.security.NoSuchAlgorithmException;

import java.security.spec.InvalidKeySpecException;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import su.usatu.project23.model.\*;

import su.usatu.project23.util.MySQLJDBCUtil;

import su.usatu.project23.util.PDFUtil;

import su.usatu.project23.util.PasswordUtil;

import su.usatu.project23.util.StringUtil;

public class Project23DAOImplementation implements Project23DAO {

String output = null;

@Override

public Rates getRatesById(int id) {

Rates rates = new Rates();

String sqlQuery = "SELECT \* FROM rates WHERE rates\_set\_id = " + id + ";";

try (Connection conn = MySQLJDBCUtil.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sqlQuery);) {

while (rs.next()) {

rates.id = rs.getInt("rates\_set\_id");

rates.single\_glazing\_price = rs.getDouble("single\_glazing\_price");

rates.double\_glazing\_price = rs.getDouble("double\_glazing\_price");

rates.triple\_glazing\_price = rs.getDouble("triple\_glazing\_price");

break;

}

} catch (SQLException ex) {

output = ex.getMessage();

System.out.println(ex.getMessage());

}

return rates;

}

@Override

public boolean editRates(String token, Rates rates) {

try {

User user = new User();

user = getUserByToken(token, "users");

String sqlUpdate = "UPDATE rates SET single\_glazing\_price = ?, double\_glazing\_price = ?, triple\_glazing\_price = ?, updated\_at = UNIX\_TIMESTAMP(), updated\_by = ? WHERE rates\_set\_id = ?";

Connection conn = MySQLJDBCUtil.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sqlUpdate);

pstmt.setDouble(1, rates.single\_glazing\_price);

pstmt.setDouble(2, rates.double\_glazing\_price);

pstmt.setDouble(3, rates.triple\_glazing\_price);

pstmt.setInt(4, user.getId());

pstmt.setInt(5, rates.id);

pstmt.executeUpdate();

pstmt.close();

return true;

} catch (SQLException e) {

output = e.getMessage();

e.printStackTrace();

return false;

}

}

@Override

public boolean addUser(User user, String tableName) {

try {

String sqlInsert = "INSERT INTO users (id, username, password, salt, email, full\_name, created\_at, group\_id, api\_token, meter\_mode, rates\_set\_id) VALUES (NULL,?,?,?,?,?,UNIX\_TIMESTAMP(),?,?,?,?)";

Connection conn = MySQLJDBCUtil.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sqlInsert);

pstmt.setString(1, user.getUsername());

pstmt.setString(2, user.getPassword());

pstmt.setString(3, user.getSalt());

pstmt.setString(4, user.getEmail());

pstmt.setString(5, user.getFullName());

pstmt.setInt(6, 2);

pstmt.setString(7, user.getApiToken());

pstmt.setInt(8, 0);

pstmt.setInt(9, 0);

pstmt.executeUpdate();

pstmt.close();

return true;

} catch (SQLException e) {

output = e.getMessage();

e.printStackTrace();

return false;

}

}

@Override

public User getUserByToken(String token, String tableName) {

User user = new User();

// set guest credentials by default

user.setId(1);

user.setUsername("guest");

user.setGroupId(3);

String sqlQuery = "SELECT \* FROM " + tableName + " WHERE api\_token = '" + token + "';";

try (Connection conn = MySQLJDBCUtil.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sqlQuery);) {

while (rs.next()) {

user.setId(rs.getInt("id"));

user.setUsername(rs.getString("username"));

user.setPassword(rs.getString("password"));

user.setSalt(rs.getString("salt"));

user.setEmail(rs.getString("email"));

user.setFullName(rs.getString("full\_name"));

user.setGroupId(rs.getInt("group\_id"));

user.setApiToken(rs.getString("api\_token"));

user.setMeterMode(rs.getInt("meter\_mode"));

user.setRatesSetId(rs.getInt("rates\_set\_id"));

break;

}

} catch (SQLException ex) {

output = ex.getMessage();

System.out.println(ex.getMessage());

}

return user;

}

@Override

public User getUserByUsername(String username, String tableName) {

User user = new User();

// set guest credentials by default

user.setId(1);

user.setUsername("guest");

user.setGroupId(3);

String sqlQuery = "SELECT \* FROM " + tableName + " WHERE username = '" + username + "';";

try (Connection conn = MySQLJDBCUtil.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sqlQuery);) {

while (rs.next()) {

user.setId(rs.getInt("id"));

user.setUsername(rs.getString("username"));

user.setPassword(rs.getString("password"));

user.setSalt(rs.getString("salt"));

user.setEmail(rs.getString("email"));

user.setFullName(rs.getString("full\_name"));

user.setGroupId(rs.getInt("group\_id"));

user.setApiToken(rs.getString("api\_token"));

user.setMeterMode(rs.getInt("meter\_mode"));

user.setRatesSetId(rs.getInt("rates\_set\_id"));

break;

}

} catch (SQLException ex) {

output = ex.getMessage();

System.out.println(ex.getMessage());

}

return user;

}

@Override

public User getUserInfoByToken(String token, String tableName) {

User user = new User();

// set guest credentials by default

user.setId(1);

user.setUsername("guest");

user.setGroupId(3);

String sqlQuery = "SELECT \* FROM " + tableName + " WHERE api\_token = '" + token + "';";

try (Connection conn = MySQLJDBCUtil.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sqlQuery);) {

while (rs.next()) {

user.setId(rs.getInt("id"));

user.setUsername(rs.getString("username"));

user.setEmail(rs.getString("email"));

user.setFullName(rs.getString("full\_name"));

user.setGroupId(rs.getInt("group\_id"));

user.setApiToken(rs.getString("api\_token"));

user.setMeterMode(rs.getInt("meter\_mode"));

user.setRatesSetId(rs.getInt("rates\_set\_id"));

break;

}

} catch (SQLException ex) {

output = ex.getMessage();

System.out.println(ex.getMessage());

}

return user;

}

@Override

public boolean updateUser(String token, User user) {

try {

String sqlUpdate = "UPDATE users SET password = ?, salt = ?, email = ?, full\_name = ?, api\_token = ? WHERE api\_token = '"

+ token + "';";

Connection conn = MySQLJDBCUtil.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sqlUpdate);

pstmt.setString(1, user.getPassword());

pstmt.setString(2, user.getSalt());

pstmt.setString(3, user.getEmail());

pstmt.setString(4, user.getFullName());

pstmt.setString(5, user.getApiToken());

pstmt.executeUpdate();

pstmt.close();

return true;

} catch (SQLException e) {

output = e.getMessage();

e.printStackTrace();

return false;

}

}

@Override

public boolean checkDbValueIfUnique(String rowLabel, String value, String tableName) {

boolean result = true;

String sqlQuery = "SELECT \* FROM " + tableName + " WHERE " + rowLabel + " = '" + value + "'";

try (Connection conn = MySQLJDBCUtil.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sqlQuery);) {

while (rs.next()) {

result = false;

break;

}

} catch (SQLException ex) {

result = false;

output = ex.getMessage();

System.out.println(ex.getMessage());

}

return result;

}

@Override

public boolean checkLoginPasswordMatch(User user, String tableName) {

boolean result = false;

String sqlQuery = "SELECT \* FROM " + tableName + " WHERE username = '" + user.getUsername() + "'";

String userPasswordFromDb = null;

String userSaltFromDb = null;

try (Connection conn = MySQLJDBCUtil.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sqlQuery);) {

// loop through the result set

while (rs.next()) {

userPasswordFromDb = rs.getString("password");

userSaltFromDb = rs.getString("salt");

try {

result = PasswordUtil.checkPassword(user.getPassword(), userSaltFromDb, userPasswordFromDb);

} catch (NoSuchAlgorithmException | InvalidKeySpecException e) {

output = e.getMessage();

e.printStackTrace();

}

break;

}

} catch (SQLException ex) {

output = ex.getMessage();

System.out.println(ex.getMessage());

}

return result;

}

@Override

public String createPdfReport(ReportData dataForPDF) throws IllegalStateException, IOException {

final String WWW\_DIR = "/srv/nginx/";

final String CONTENT\_DIR = "/user-content/";

final String FONTS\_DIR = WWW\_DIR + CONTENT\_DIR + "/fonts/";

final String IMG\_DIR = WWW\_DIR + CONTENT\_DIR + "/img/";

byte[] array = new byte[16]; // length is bounded by 16

new Random().nextBytes(array);

String newPdfName = StringUtil.generateRandomString();

String userAccessPath = CONTENT\_DIR + newPdfName + ".pdf";

String savingPath = WWW\_DIR + userAccessPath;

if (PDFUtil.generateNewPDF(dataForPDF, FONTS\_DIR, IMG\_DIR, savingPath)) {

return userAccessPath;

} else {

return "Failed";

}

}

@Override

public boolean assignPdfReportToUser(String token, String documentName) {

try {

String sqlInsert = "INSERT INTO reports (id, owner\_id, document\_name, created\_at) VALUES (NULL, ?, ?, UNIX\_TIMESTAMP());";

Connection conn = MySQLJDBCUtil.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sqlInsert);

User user = getUserByToken(token, "users");

pstmt.setInt(1, user.getId());

pstmt.setString(2, documentName);

pstmt.executeUpdate();

pstmt.close();

return true;

} catch (SQLException e) {

output = e.getMessage();

e.printStackTrace();

return false;

}

}

@Override

public String[] getUserPdfFiles(int userId) {

String sqlQuery = "SELECT \* FROM reports WHERE owner\_id = '" + userId + "';";

String[] myArray = null;

List<String> list = new ArrayList<String>();

try (Connection conn = MySQLJDBCUtil.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sqlQuery);) {

while (rs.next()) {

list.add(rs.getString("document\_name"));

}

myArray = new String[list.size()];

list.toArray(myArray);

} catch (SQLException ex) {

System.out.println(ex.getMessage());

}

return myArray;

}

@Override

public boolean deletePdfReport(String token, String documentName) {

boolean deletionStatus = false;

User user = new User();

user = getUserByToken(token, "users");

int userId = user.getId();

String sqlUpdate = "DELETE FROM reports WHERE document\_name = '" + documentName + "' AND owner\_id = " + userId

+ ";";

try (Connection conn = MySQLJDBCUtil.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sqlUpdate)) {

int rowAffected = pstmt.executeUpdate();

if (rowAffected == 1)

deletionStatus = true;

} catch (SQLException ex) {

output = ex.getMessage();

System.out.println(ex.getMessage());

}

return deletionStatus;

}

}

**JsonResponse**

package su.usatu.project23.model;

public class JsonResponse {

public boolean success;

public String errorMessage;

public Object responseBody;

}

**Rates**

package su.usatu.project23.model;

public class Rates {

public int id; //идентификатор группы тарифов

public double single\_glazing\_price;

public double double\_glazing\_price;

public double triple\_glazing\_price;

}

**ReportData**

package su.usatu.project23.model;

public class ReportData {

public String width = "—";

public String height = "—";

public String sashesCount = "—";

public String glazing = "—";

public String totalAmount = "—";

}

**User**

package su.usatu.project23.model;

public class User {

private int id;

private String username;

private String password;

private String salt;

private String email;

private String fullName;

private int groupId;

private String apiToken;

private int meterMode;

private int ratesSetId;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getSalt() {

return salt;

}

public void setSalt(String salt) {

this.salt = salt;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getFullName() {

return fullName;

}

public void setFullName(String lastName) {

this.fullName = lastName;

}

public int getGroupId() {

return groupId;

}

public void setGroupId(int groupId) {

this.groupId = groupId;

}

public String getApiToken() {

return apiToken;

}

public void setApiToken(String apiToken) {

this.apiToken = apiToken;

}

public int getMeterMode() {

return meterMode;

}

public void setMeterMode(int meterMode) {

this.meterMode = meterMode;

}

public int getRatesSetId() {

return ratesSetId;

}

public void setRatesSetId(int ratesSetId) {

this.ratesSetId = ratesSetId;

}

}

**Util**

**JsonResponseUtil**

package su.usatu.project23.util;

import com.google.gson.Gson;

import su.usatu.project23.model.JsonResponse;

public class JsonResponseUtil {

public JsonResponseUtil() {

}

// Form response on failure

public static String formJsonResponse(String status, String message) {

JsonResponse jsonObject = new JsonResponse();

jsonObject.success = false;

if (status == "failure") {

jsonObject.errorMessage = message;

} else {

jsonObject.errorMessage = "Unknown JsonResponse status name";

}

String jsonString = new Gson().toJson(jsonObject);

return jsonString;

}

// Form response on success

public static String formJsonResponse(String status, String message, Object objectToPass) {

JsonResponse jsonObject = new JsonResponse();

if (status == "success") {

jsonObject.success = true;

jsonObject.errorMessage = message;

jsonObject.responseBody = objectToPass;

} else {

jsonObject.success = false;

jsonObject.errorMessage = "Unknown JsonResponse status name";

}

String jsonString = new Gson().toJson(jsonObject);

return jsonString;

}

}

**MySQLJDBCUtil**

package su.usatu.project23.util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class MySQLJDBCUtil {

public static Connection getConnection() throws SQLException {

Connection conn = null;

String url = "jdbc:mysql://mysql:3306/project23";

String user = "db\_user\_23";

String password = "db\_password\_23";

conn = DriverManager.getConnection(url, user, password);

return conn;

}

}

**PasswordUtil**

package su.usatu.project23.util;

import java.util.Base64;

import java.security.NoSuchAlgorithmException;

import java.security.SecureRandom;

import java.security.spec.InvalidKeySpecException;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.PBEKeySpec;

import java.math.BigInteger;

public class PasswordUtil {

private static final SecureRandom RANDOM = new SecureRandom();

private static final int ITERATIONS = 1000;

private static final int KEY\_LENGTH = 192; // bits

public static String getSalt() {

byte[] salt = new byte[16];

RANDOM.nextBytes(salt);

return Base64.getEncoder().encodeToString(salt);

}

public static String hashPassword(String password, String salt)

throws NoSuchAlgorithmException, InvalidKeySpecException {

char[] passwordChars = password.toCharArray();

byte[] saltBytes = salt.getBytes();

PBEKeySpec spec = new PBEKeySpec(passwordChars, saltBytes, ITERATIONS, KEY\_LENGTH);

SecretKeyFactory key = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");

byte[] hashedPassword = key.generateSecret(spec).getEncoded();

return String.format("%x", new BigInteger(hashedPassword));

}

public static boolean checkPassword(String password, String salt, String expectedHash)

throws NoSuchAlgorithmException, InvalidKeySpecException {

String hashToCheck = hashPassword(password, salt);

if (hashToCheck.length() != expectedHash.length())

return false;

if (!hashToCheck.equals(expectedHash))

return false;

return true;

}

}

**PDFUtil**

package su.usatu.project23.util;

import java.io.File;

import java.io.IOException;

import java.time.ZoneId;

import java.time.ZonedDateTime;

import java.time.format.DateTimeFormatter;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.pdmodel.PDPage;

import org.apache.pdfbox.pdmodel.PDPageContentStream;

import org.apache.pdfbox.pdmodel.common.PDRectangle;

import org.apache.pdfbox.pdmodel.font.PDFont;

import org.apache.pdfbox.pdmodel.font.PDType0Font;

import org.apache.pdfbox.pdmodel.graphics.image.PDImageXObject;

import org.vandeseer.easytable.TableDrawer;

import org.vandeseer.easytable.structure.Row;

import org.vandeseer.easytable.structure.Table;

import org.vandeseer.easytable.structure.cell.TextCell;

import su.usatu.project23.model.ReportData;

public class PDFUtil {

public static boolean generateNewPDF(ReportData dataForPDF, String fontsPath, String imgPath, String savingPath)

throws IOException, IllegalStateException {

boolean pdfGenerationStatus = false;

try (PDDocument document = new PDDocument()) {

final PDPage page = new PDPage(PDRectangle.A4);

document.addPage(page);

try (PDPageContentStream contentStream = new PDPageContentStream(document, page)) {

// image

PDImageXObject image = PDImageXObject.createFromFile(imgPath + "favicon.png", document);

contentStream.drawImage(image, 25, 790);

// text

contentStream.beginText();

PDFont timesRegular = PDType0Font.load(document, new File(fontsPath + "times.ttf"));

PDFont timesBold = PDType0Font.load(document, new File(fontsPath + "timesbd.ttf"));

PDFont timesItalic = PDType0Font.load(document, new File(fontsPath + "timesi.ttf"));

contentStream.setFont(timesBold, 20);

contentStream.setLeading(14 \* 1.25f);

contentStream.newLineAtOffset(0, 828);

contentStream.newLineAtOffset(82, -20);

String line1 = "Отчёт";

contentStream.showText(line1);

contentStream.newLine();

contentStream.setFont(timesRegular, 14);

String line2 = "по произведённому расчёту стоимости производства оконных конструкций";

contentStream.showText(line2);

contentStream.newLine();

ZonedDateTime currDateObj = ZonedDateTime.now(ZoneId.of("Asia/Yekaterinburg"));

DateTimeFormatter dateFormatObj = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

String formattedDate = currDateObj.format(dateFormatObj);

// сбрасываем координаты для newLineAtOffset

contentStream.endText();

contentStream.beginText();

contentStream.newLineAtOffset(265, 25);

contentStream.setFont(timesItalic, 14);

String dateLine = "Дата формирования отчёта: " + formattedDate;

contentStream.showText(dateLine);

contentStream.newLine();

contentStream.endText();

// table

// Build calculationTable

Table calculationTable = Table.builder().addColumnsOfWidth(150, 150).font(timesRegular).padding(2)

.addRow(Row.builder()

.add(TextCell.builder().text("Варианты створок").borderWidth(1).build())

.add(TextCell.builder().text(dataForPDF.sashesCount).borderWidth(1).build())

.build())

.addRow(Row.builder()

.add(TextCell.builder().text("Стеклопакет").borderWidth(1).build())

.add(TextCell.builder().text(dataForPDF.glazing).borderWidth(1).build())

.build())

.addRow(Row.builder()

.add(TextCell.builder().text("Высота").borderWidth(1).build())

.add(TextCell.builder().text(dataForPDF.width).borderWidth(1).build())

.build())

.addRow(Row.builder()

.add(TextCell.builder().text("Ширина").borderWidth(1).build())

.add(TextCell.builder().text(dataForPDF.height).borderWidth(1).build())

.build())

.addRow(Row.builder()

.add(TextCell.builder().text("Итого").borderWidth(1).font(timesItalic).build())

.add(TextCell.builder().text(dataForPDF.totalAmount).borderWidth(1).build())

.build())

.build();

// Set up the drawer for calculationTable

TableDrawer calculationTableDrawer = TableDrawer.builder().contentStream(contentStream).startX(25)

.startY(page.getMediaBox().getUpperRightY() - 80).table(calculationTable).build();

// Draw the tables

calculationTableDrawer.draw();

}

document.save(savingPath);

document.close();

pdfGenerationStatus = true;

}

return pdfGenerationStatus;

}

}

**StringUtil**

package su.usatu.project23.util;

import java.util.Random;

// Source: https://www.programiz.com/java-programming/examples/generate-random-string

public class StringUtil {

public static String generateRandomString() {

// create a string of uppercase and lowercase characters and numbers

String upperAlphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

String lowerAlphabet = "abcdefghijklmnopqrstuvwxyz";

String numbers = "0123456789";

// combine all strings

String alphaNumeric = upperAlphabet + lowerAlphabet + numbers;

// create random string builder

StringBuilder sb = new StringBuilder();

// create an object of Random class

Random random = new Random();

// specify length of random string

int length = 24;

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

// generate random index number

int index = random.nextInt(alphaNumeric.length());

// get character specified by index

// from the string

char randomChar = alphaNumeric.charAt(index);

// append the character to string builder

sb.append(randomChar);

}

String randomString = sb.toString();

return randomString;

}

}

**TokenUtil**

package su.usatu.project23.util;

import java.security.SecureRandom;

import java.util.Base64;

public class TokenUtil {

private static final SecureRandom secureRandom = new SecureRandom(); // threadsafe

private static final Base64.Encoder base64Encoder = Base64.getUrlEncoder(); // threadsafe

public static String generateNewToken() {

byte[] randomBytes = new byte[24];

secureRandom.nextBytes(randomBytes);

return base64Encoder.encodeToString(randomBytes);

}

}

**Приложение 3**

**Unit тесты**

**JsonResponseUtilTest**

package su.usatu.project23.test;

import org.junit.Test;

import static org.junit.Assert.assertFalse;

import com.google.gson.Gson;

import su.usatu.project23.model.JsonResponse;

import su.usatu.project23.util.JsonResponseUtil;

public class JsonResponseUtilTest {

@Test

public void testInvalidJson() {

String invalidJsonResponse = JsonResponseUtil.formJsonResponse("success", "message");

Gson gson = new Gson();

JsonResponse jr = gson.fromJson(invalidJsonResponse, JsonResponse.class);

assertFalse(jr.success);

}

}

**Приложение 3**

**Unit тесты**

**RandomStringTest**

package su.usatu.project23.test;

import static org.junit.Assert.assertFalse;

import static org.junit.Assert.assertTrue;

import org.junit.Test;

import su.usatu.project23.util.StringUtil;

public class RandomStringTest {

@Test

public void StringLengthTest() {

String testString = StringUtil.generateRandomString();

assertTrue(testString.length() == 24);

String secondTestString = StringUtil.generateRandomString();

assertTrue(secondTestString.length() == 24);

assertFalse(testString.equals(secondTestString));

}

}

**Приложение 5**

**Unit тесты**

**UserModelTest**

package su.usatu.project23.test;

import static org.junit.Assert.assertNotNull;

import org.junit.BeforeClass;

import org.junit.Test;

import su.usatu.project23.model.User;

public class UserModelTest {

static User user;

@BeforeClass

public static void beforeClass() {

user = new User();

}

@Test

public void userGetSetTest() {

user.setId(10);

assertNotNull(user.getId());

user.setUsername("username");

assertNotNull(user.getUsername());

user.setPassword("password");

assertNotNull(user.getPassword());

user.setSalt("stringWithSalt");

assertNotNull(user.getSalt());

user.setEmail("example@example.com");

assertNotNull(user.getEmail());

user.setFullName("John Doe");

assertNotNull(user.getFullName());

user.setGroupId(10);

assertNotNull(user.getGroupId());

user.setApiToken("token");

assertNotNull(user.getApiToken());

user.setMeterMode(3);

assertNotNull(user.getMeterMode());

user.setRatesSetId(2);

assertNotNull(user.getRatesSetId());

}

}

**Приложение 6**

**Unit тесты**

**Test**

package su.usatu.project23.test;

import org.junit.Test;

import static org.junit.Assert.assertFalse;

import com.google.gson.Gson;

import su.usatu.project23.model.JsonResponse;

import su.usatu.project23.util.JsonResponseUtil;

public class JsonResponseUtilTest {

@Test

public void testInvalidJson() {

String invalidJsonResponse = JsonResponseUtil.formJsonResponse("success", "message");

Gson gson = new Gson();

JsonResponse jr = gson.fromJson(invalidJsonResponse, JsonResponse.class);

assertFalse(jr.success);

}

}

|  |
| --- |
| **Приложение 7**  **РУКОВОДСТВО**  **ПОЛЬЗОВАТЕЛЯ**  Калькулятор расчёта стоимости оконных конструкций  «Калькулятор расчёта стоимости оконных конструкций»  Инв. № подл.  Подп. И дата  Взаим.инв.№  Инв. № дубл.  Подп. и дата  ***Пта*** |
| |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | | | | | | |  |  |  |  |  | | ***Изм.*** | ***Лист*** | ***№докум.*** | ***Подп.*** | ***Дата*** | | ***Разраб.*** | |  |  |  | ***Калькулятор расчёта стоимости оконных конструкций.*** | ***Лит.*** | | | ***Лист*** | ***Листов*** | | ***Пров.*** | |  |  |  |  |  |  |  |  | |  | |  |  |  | ***ФГБОУ ВО «УГАТУ»*** | | | | | | ***Н.контр.*** | |  |  |  | | ***Утв.*** | |  |  |  |   Документ является руководством пользователя для системы «Калькулятор стоимости производства оконных конструкций»  Документ разработан в рамках курсовой работы по теме: «разработка кроссплатформенного программного продукта на языке JAVA с использованием системы контроля версий».  **Аннотация**  Инв. № подл.  Подп. И дата  Взаим.инв.№  Инв. № дубл.  Подп. и дата  ***Пта*** |
| 1 Назначение программы……………………………………………………4  2 Условия выполнения программы ………………………………………..5  3 Выполнение программы…………………………………………………..6  Инв. № подл.  Подп. И дата  Взаим.инв.№  Инв. № дубл.  Подп. и дата  ***Пта***     |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | Лист3 | |  |  |  |  |  | | ***Изм.*** | ***Лист*** | ***№докум.*** | ***Подп.*** | ***Дата*** | |
| Инв. № подл.  Подп. И дата  Взаим.инв.№  Инв. № дубл.  Подп. и дата  ***Пта***  **1. Назначение программы**  Программа представляет собой веб-приложение, которое используется для расчётов стоимости производства оконных конструкций для промышленных предприятий   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | Лист4 | |  |  |  |  |  | | ***Изм.*** | ***Лист*** | ***№докум.*** | ***Подп.*** | ***Дата*** | |
| **2. Условия выполнения**  Для выполнения работы веб-приложения необходимо иметь устройство с доступом в Интернет (например, ПК, смартфон, планшетный компьютер), на которых установлен веб-браузер с поддержкой технологий Cookie и JavaScript. Такими веб-браузерами, к примеру, являются Mozilla Firefox 88 и Google Chrome 90.  Инв. № подл.  Подп. И дата  Взаим.инв.№  Инв. № дубл.  Подп. и дата  ***Пта***     |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | Лист5 | |  |  |  |  |  | | ***Изм.*** | ***Лист*** | ***№докум.*** | ***Подп.*** | ***Дата*** | |
| **3. Выполнение программы**  Все действия выполняются на сайте, имеющий адрес <https://project23.usatu.su/>  **3.1. *Регистрация и вход:***  В случае первого посещения сайта перейдите в раздел «Зарегистрироваться» и введите ваши будущие данные, после чего войдите в аккаунт через раздел «Войти в аккаунт», введя данные при регистрации. (Рис.1).    Рис.1 – Поля регистрации  Инв. № подл.  Подп. И дата  Взаим.инв.№  Инв. № дубл.  Подп. и дата  ***Пта***     |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | Лист6 | |  |  |  |  |  | | ***Изм.*** | ***Лист*** | ***№докум.*** | ***Подп.*** | ***Дата*** | |
| Инв. № подл.  Подп. И дата  Взаим.инв.№  Инв. № дубл.  Подп. и дата  ***Пта***  **3.2. *Калькулятор:***  Для выполнения программы нужно перейти на сайт и авторизоваться. После чего нажать «Рассчитать оплату», выбрать нужный тариф и заполнить поля в соответствии с тарифом (Рис.2).    Рисунок 2. «Калькулятор стоимости производства оконных конструций»  3.3. Расчёт стоимости производства в PDF  В случае необходимости отчёта о стоимости производства, вы можете нажать на кнопку «PDF-отчёт» для формирования PDF файла с отчётом (Рис.3).     |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | Лист7 | |  |  |  |  |  | | ***Изм.*** | ***Лист*** | ***№докум.*** | ***Подп.*** | ***Дата*** | |
| Инв. № подл.  Подп. И дата  Взаим.инв.№  Инв. № дубл.  Подп. и дата  ***Пта***  Изображение выглядит как стол  Автоматически созданное описание  Рисунок 3. Отчёт  3.4. Изменение коэффицентов:  Администратор может изменять коэффиченты расчёта стоимости оконных конструкций, затем изменения зафиксируются в программе (Рис.4).  Изображение выглядит как текст  Автоматически созданное описание  Рисунок 4. Изменение коэффицентов     |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | Лист8 | |  |  |  |  |  | | ***Изм.*** | ***Лист*** | ***№докум.*** | ***Подп.*** | ***Дата*** | |

**Список литературы**

1. ГОСТ 12506-81. Окна деревянные для производственных зданий. Типы, конструкция и размеры. [Москва], 1984. URL: http://do cs.cntd.ru/document/gost-12506-81 (дата обращения: 21.02.2021)

2. Окна для промышленных зданий. [Санкт-Петербург], 2006-2021. URL: https://okna-stroyvector.ru/stroitelnym-kompaniyam/ok.. (дата обращения: 23.02.2021)

3. ГОСТы и СНиПы по стеклопакетам. [Москва], 2020. URL: http://citi okna.ru/produkciya/steklopakety/gost.html (дата обращения: 21.02.2021) – Т

4. Справочник строителя | Окна для производственных зданий, [Электронный ресурс] 2007-2020. - URL: https://baurum.ru/\_library/?cat=win dows-wooden-produc.. (дата обращения: 21.02.2021)

5. ФЗ О защите прав потребителей: ФЗ от 07.02.1992 N 2300-1 (ред. от 08.12.2020) (с изм. и.дом.) [Электронный ресурс] URL: http://www.consultant.ru/document/cons\_doc\_LAW\_305/ (дата обращения: 20.02.2021).

6. Сукиасян, Э.Р. Список литературы к курсовой и дипломной работе. Рекомендации по составлению [Текст] / Э.Р. Сукиасян. – Москва, 2001.