Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Инновационного непрерывного образования

Кафедра Информатики

К защите допустить:

Заведующий кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Волорова

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

система нормирования материалов в конструкторской документации

БГУИР ДП 1-40 04 01 00 103 ПЗ

Студент А.В. Авхимович

Руководитель А.В. Жвакина

Консультанты:

от кафедры информатики А.С. Летохо

по экономической части Д.А. Савчик

Нормоконтролер В.В. Шиманский

Рецензент

Минск 2019

Реферат

Дипломный проект представлен следующим образом. Чертежи и плакаты: 6 листов формата А1. Пояснительная записка: 42 страниц, 30 рисунков, 4 таблицы, 6 литературных источников, 1 приложение.

Ключевые слова: КЛИЕНТСКАЯ ЧАСТЬ, СЕРВЕРНАЯ ЧАСТЬ, БАЗА ДАННЫХ

Целью дипломного проекта является разработка Системы нормирования материалов по конструкторской документации для отдела технологического инжиниринга машиностроительного завода, которая позволяет удобно вести базу данных материалов, график посчитаных(пронармированных) узлов изделий, генерировать отчеты в формате данных Microsoft Excel, и другие возможности.

Для достижения цели дипломного проекта было разработано клиент-серверное приложение, а также подобраны и настроены другие программные средства, необходимые для работы системы.

При разработке и внедрении приложения использовался такой стек технологий как CSS, MySQL.

В разделе введения описано назначение разрабатываемого ПС.

В первой главе описан обзор аналогов программного продукта, а также сформированы требований к проектируемому ПС.

Во второй главе определены требования к программе, а также цели и задачи проектирования.

В третьей главе представлен процесс проектирования, используемые технологии и средства разработки, а такде выбранная архитектура приложения.

В четрвертой главе рассмотрен процесс разработки программного средства: приведена структура проекта, а также классы моделеей, контроллеров и представлений.

В пятой главе приведено описание использования ПС.

В шестой главе приведено технико-экономическое обоснование, произведен расчет затрат организации-подрядчика на выполнение заказываемых работ по проекту, рассчитана прибыль и рентабельность для организации-подрядчика.

Заключение содержит выводы по данному дипломному проекту и результаты его выполнения и внедрения.

Дипломный проект является завершенным, поставленная задача решена в полной мере.

Министерство образования Республики Беларусь

Учреждение образования

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Факультет | ИНО | | | Кафедра | | | Информатики | | | | | | | | | |
| Специальность | 1-40 04 01 | | | Специализация | | | | | 00 | | | | | | | |
| УТВЕРЖДАЮ | | | | | | | | | | | | | | | | |
|  | | | | | | | |  | | | | | Н.А. Волорова | | | |
| « | | | | | | | |  | | » |  | | | | 2019 г. | | |
| **ЗАДАНИЕ** | | | | | | | | | | | | | | | | |
| **по дипломному проекту студента** | | | | | | | | | | | | | | | | |
| Авхимовича Алексея Валерьевича | | | | | | | | | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | | | | | | | | | |
| 1. Тема проекта: | | Система нормирования материалов в конструкторской документации | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| утверждена приказом по университету от | | | | « | 28 | » | | апреля | | | | 2019 г. | | № | | 692-с |
| 2. Срок сдачи студентом законченной работы | | | | |  | | | | | | | | | | | |
| 3. Исходные данные к проекту | | | Тип программного средства – Cистема; | | | | | | | | | | | | | |
| Языки программирования – Java. | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Назначение разработки: дать возможность пользователям программного средства | | | | | | | | | | | | | | | | |
| управлять взаимоотношениями с клиентами. | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| 4. Содержание пояснительной записки (перечень подлежащих разработке вопросов) | | | | | | | | | | | | | | | | |
| Введение | | | | | | | | | | | | | | | | |
| 1.АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ | | | | | | | | | | | | | | | | |
| 2. ПОСТАНОВКА ЗАДАЧИ | | | | | | | | | | | | | | | | |
| 3. ПРОЕКТИРОВАНИЕ ПС | | | | | | | | | | | | | | | | |
| 4. РАЗРАБОТКА ПС | | | | | | | | | | | | | | | | |
| 5. ОПИСАНИЕ ИСПОЛЬЗОВАНИЯ | | | | | | | | | | | | | | | | |
| 6. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА ПО РАЗРАБОТКЕ ПС | | | | | | | | | | | | | | | | |
| ЗАКЛЮЧЕНИЕ | | | | | | | | | | | | | | | | |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | | | | | | | | | | | | | | | | |
| ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММНОГО КОДА | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. Перечень графического материала (с точным указанием наименования) и обозначения | | | |
| вида и типа материала) | |  | |
| Схема структуры программы – формат А1, лист 1 | | | |
| Диаграмма классов – формат А1, лист 1 | | | |
| Схема базы данных – формат А1, лист 1 | | | |
| Миграция связанных записей. Схема программы – формат А1, лист 1 | | | |
| Диаграмма потоков данных системы. Плакат – формат А1, лист 1 | | | |
| Визуальный интерфейс программного средства. Плакат – формат А1, лист 1 | | | |
|  | | | |
|  | | | |
| 6. Содержание задания по технико-экономическому обоснованию | | | |
| Расчет затрат на разработку, прибыли и рентабельности проекта для | | | |
| компании-подрядчика. | | | |
| Задание выдал |  | | Е.Е. Марченкова |
|  | | | |

**КАЛЕНДАРНЫЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов дипломного проекта (работы) | Объем этапа в % | Срок выполнения этапа | Приме­чание |
| Анализ предметной области, разработка технического задания, выбор подходящей основы для реализации | 20 | 28.04 – 06.05 |  |
| Проектирование архитектуры приложения | 10 | 07.05 – 11.05 |  |
| Разработка модулей | 20 | 12.05 – 20.05 |  |
| Разработка API и миграция данных | 20 | 21.05 – 29.05 |  |
| Расчет экономической эффективности | 10 | 30.05 – 04.06 |  |
| Оформление графического материала и пояснительной записки | 20 | 05.06 – 14.06 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | 28 апреля 2019 г. | | Руководитель | |  | А.В. Жвакина |
| Задание принял к исполнению | |  | | А.В. Авхимович | | |

Содержание

[ВВЕДЕНИЕ 6](#_Toc8247169)

[**1.** **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ** 7](#_Toc8247170)

[**1.1** **ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ** 7](#_Toc8247171)

[**1.2** **Обзор аналогов программного продукта** 8](#_Toc8247172)

[**1.3** **Формирование требований к проектируемому ПС** 9](#_Toc8247173)

[2. Постановка задачи 11](#_Toc8247174)

[2.1 Требования к программе 11](#_Toc8247175)

[2.2 Цели и задачи проектирования 11](#_Toc8247176)

[3. Проектирование ПС 13](#_Toc8247177)

[3.1 Используемые технологии и средства разработки 13](#_Toc8247178)

[**3.1.1** **Программная платформа Java** 13](#_Toc8247179)

[**3.1.2** **NetBeans** 14](#_Toc8247180)

[**3.1.3** **JavaFX** 15](#_Toc8247181)

[**3.1.4** **MySQL** 16](#_Toc8247182)

[**3.1.5** **Библиотека Apache POI** 16](#_Toc8247183)

[3.2 Архитектура приложения 16](#_Toc8247184)

[4. Разработка ПС 19](#_Toc8247185)

[4.1 Реализация клиентского-приложения 19](#_Toc8247186)

[5. Описание использования 21](#_Toc8247187)

[5.1 Система нормирования материалов в конструкторской документации 21](#_Toc8247188)

[6. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА ПО РАЗРАБОТКЕ ПРОГРАММНОГО СРЕДСТВА 28](#_Toc8247189)

[6.1 Общая характеристика программного средства 28](#_Toc8247190)

[6.2 Расчет затрат на разработку ПО 28](#_Toc8247191)

[**6.2.1** **Расчет затрат на основную заработную плату разработчиков** 28](#_Toc8247192)

[**6.2.2** **Расчет затрат на дополнительную заработную плату** 29](#_Toc8247193)

[**6.2.3** **Расчет затрат на социальные нужды** 30](#_Toc8247194)

[**6.2.4** **Расчет прочих затрат** 30](#_Toc8247195)

[6.3 Оценка результата от продажи ПО. Расчет экономического эффекта. 31](#_Toc8247196)

[6.4 Расчет показателей эффективности инвестиций в разработку ПО 32](#_Toc8247197)

[6.5 Выводы по технико-экономическому обоснованию 35](#_Toc8247198)

[ЗАКЛЮЧЕНИЕ 36](#_Toc8247199)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 37](#_Toc8247200)

[ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММНОГО КОДА 38](#_Toc8247201)

ВВЕДЕНИЕ

На сегодняшний день трудно представить нормировани материалов технологами вручную используя Microsoft Excel.

В связи с этим будет разработано программное обеспечение, Система нормирования материалов по конструкторской документации призванное облегчить, систематизировать и автоматизировать работу технологов.

Для реализации поставленной цели необходимо выполнить следую-щие задачи:

Изучить базу материалов предприятия;

Собрать информацию о расходе материалов предприятия, такие как припуски на обработку, резку, а также определить допустимые коэффиценты материала на отход;

Рассмотреть и обобщить собранную информацию.

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ** 
   1. **Описание предметногй области**

На современном производстве одной из важнейших составляющих себестоимости изделия является норма расхода материалов, быстрый и качественный расчет которой является залогом верного расчета себестоимости изделия. Эта работа должна быть проведена максимально быстро и точно. Если норма расхода, а соответственно, и цена будут завышены, то заказчик уйдет к другому производителю, а, с другой стороны, норма если окажется заниженной, то пострадает производитель.

Болышая часть современного программного обеспечения системы технологической подготовки производства, представленного на рынке, предлагает выполиять подетальный расчет норм расхода материалов на основе разработанного технологического процесса. Но несмотря на то, что разработка технологического процесса позволяет определить все используемые в производстве детали материалы, включая вспомогательные, такие сроки разработки норм расхода оказываются слишком велики. Так как основную стоимость составляет основной материал детали, а разработка технологических процессов на каждую деталь и сборочную единицу изделия требует существенных затрат времени, целесообразным является разработка подетальных норм расхода на основании чертежей и соответствующих нормативов припусков на обработку.

Разрабатываемая программа позволит рассчитывать норму расхода и размеры необходимой заготовки для деталей, не требующих деформирующих методов обработки (ковка, гибка и т. п.). По определению, норма расхода материалов на изделия (детали) —максимально допустимое плановое количество материала на изготовление изделия (детали) при установленном качестве и условиях производства. Соответственно, для определения норм расхода по установленным нормативам возможно использование максимальных значений припусков, исходя из имеющегося на предприятии оборудования.

Исходными данными для расчета в программе будут являются габаритные размеры детали по чертежу, тип заготовки (лист, труба, круг и т. д.).

После чего формируется SQL-запрос к базе данных и формируется норма расхода материала.

Внедрение данного программного продукта позволит значительно сократить время разработки норм расхода материалов, снизить вероятность ошибок инженера и, кроме того, сократить время обучения работника бюро материальных нормативов.

* 1. **Обзор аналогов программного продукта**

В настоящий момент на рынке представлены следующие приложения с похожей функциональностью. Проведем их анализ.

1. Intermech АРМ материального нормирования.

Приложение, предназначенное для расчет материальных нормативов и формирование сводных ведомостей по материалам.

Расчет материальных нормативов

АРМ материального нормирования представляет собой узкоспециализированное автоматизированное рабочее место, посредством которого осуществляется:

Назначение вариантов заготовок, вспомогательных материалов на изделие/версию изделия с учетом входимости в различные сборочные узлы или заказы.

Автоматизация расчета заготовок и вспомогательных материалов при помощи встроенной Экспертной системы.

Выпуск извещений на изменение заготовок и вспомогательных материалов с сохранением истории изменений.

Получение сводных ведомостей норм расхода материалов на изделие, заказ, по цеху и другим условиям.

Единое информационное пространство

Единое информационное пространство между технологическими и конструкторскими службами предприятия осуществляется при помощи PDM-системы Search, которая входит в состав АРМ материального нормирования.

Электронный документооборот между различными подразделениями.

Ведение списка пользователей, которые могут работать с системой (вход в систему по паролю), обеспечение безопасности путем назначения пользователям прав доступа на выполнение тех или иных действий.

Получение выборок изделий и техпроцессов по разнообразным критериям.

Система IMBase обеспечивает ведение единой конструкторско-технологической базы данных. Широкий спектр возможностей по сопровождению информационной базы позволяет адаптировать систему к условиям конкретного предприятия. Настройка прав доступа к любому элементу системы обеспечивает требуемый уровень безопасности для конкретного пользователя.

База данных по технологическому назначению содержит следующую информацию:

Применяемые основные и вспомогательные материалы.

Виды заготовок и их применяемость по сортаменту.

Методика расчета материального нормирования.

Комплекты форм выходных документов.

1. Нормирование материалов

Программа Нормирование материалов предназначается для назначения заготовки детали, автоматизации расчета нормы расхода основного материала.

При расчете заготовки учитываются припуски на обработку, некратность размеров поставляемого материала и другие нормативы технологических потерь. В зависимости от вида и профиля заготовки пользователю предоставляется возможность выбрать необходимый вид расчета. Для оптимизации расхода материалов предусмотрена возможность выполнения нескольких вариантов расчета с выбором оптимального. Предыдущие варианты расчетов сохраняются в документе.

В базовой поставке системы настроены алгоритмы расчета заготовок, получаемых резкой проката и горячей ковкой на молотах и прессах, а также горячей штамповкой на молотах.

Встроенный редактор расчетов обеспечивает ввод пользователем алгоритмов нормирования материалов, используемых на предприятии.

Программный интерфейс обеспечивает интеграцию с различными системами.

В составе комплекса автоматизации конструкторско-технологической подготовки Система интегрирована:

* со Справочником Материалы и Сортаменты (и Материалы и Сортаменты для КОМПАС) для получения необходимых данных по применяемым на предприятии материалам и сортаментам, включая размеры профиля и свойства материала;
* с системой ЛОЦМАН:PLM для формирования технологических ведомостей по расходу материалов, передачи данных в различные системы подготовки, планирования и управления производством класса MRP II\ERP;
* с САПР ТП ВЕРТИКАЛЬ.[5]
  1. **Формирование требований к проектируемому ПС**

Главной целью при разработке приложения будет являться реализация простого, понятного и удобного пользовательского интерфейса, с использованием технологии JavaFX от компании Oracle.

Основной упор будет делаться на качественную поддержку отслеживания посчитанных узлов изделия, формированию понятных и доходчивых сводных ведомостей материалов. Помимо этого, должны быть реализованы следующие возможности:

* Возможность редактирования справочник Формул;
* Возможность редактирования справочник Материалов;
* Возможность редактирования справочник Обозначений;
* Возможность редактирования справочник Вспомогательных материалов;
* График учёта посчитанных узлов изделия;
* Представление узла изделия в виде спецификации
* Древовидное представление изделия
* Возможность поиска входимости узла;
* Возможность формирования отчета сводной ведомости материалов в формате данных Excel
* Возможность формирования данных в формате Excel для проверки корректности введенных данных с конструкторской документации
* Возможность формирования данных в формате Excel для раскроя материалов
* Возможность представление узла в виде спецификации
* Возможность задания параметров для расчет материала
* Возможность задания параметров для расчет вспомогательного материала

Анализ предметной области показал, что аналоговые программные

средства имеет ряд недостатков таких как, неудобный пользовательский интерфейс, нет возможности вести учет(график) посчитанных узлов изделия, отсутствует возможность замены продублированных элементов, нет возможности сохранять сводную ведомость материалов в формате данных Microsoft Excel, а также отсутствует кроссплатформенность. В ходе разработки ПС будут устранены данные недостатки.

# Постановка задачи

## Требования к программе

Необходимо реализовать клиент-серверное приложение с использованием выбранной технологии (в данном случае был выбран фреймворк JavaFX), которая позволяет обеспечить запуск необходимых задач и выполнять соответствующую работу клиент-серверного приложения по взаимодействию с пользователем через клиентскую часть приложения.

Система нормирования материалов в конструкторской документации должна:

1. Поддерживать одновременно работу 5-10 пользователей(размер отдела технологов);
2. Поддерживать операционные системы: Windows xp, Windows 7, Windows 8, Windows 10, Linux.

## Цели и задачи проектирования

**Задачи:**

1. Изучить предметную область, ознакомиться с возможностями фреймворка JavaFX при разработке приложений
2. Изучить фреймворки, используемые совместно с изучаемой технологией JavaFX
3. Создать проект Система нормирования материалов в конструкторской документации с наиболее общим функционалом.

Основной функциональностью системы нормирования материалов по конструкторсокой документации можно обозначить следующие возможности:

1. Должно иметь представление в виде:

* Спецификации;
* Дерева.

1. Добавление/удаление/редактирование элементов(обозначений) конструкторской документации.
2. Добавление/удаление/редактирование основных, вспомогательных материалов.
3. Добавление/удаление/редактирование формул.
4. Добавление/удаление/редактирование производственных заказов.
5. Поиск вхождение элемента(обозначения) наследника к элементу (обозначению) родителя.
6. График учета пронармированных узлов изделий.
7. Формирование отчёта пронумерованного узла изделия в формате данных excel.
8. Формирование данных узла изделия в формате данных excel для последующего раскроя материалов.
9. Формирование данных узла изделия в формате данных excel для проверки корректности вводимых данных.
10. Замена продублированных элементов.

В ходе постановки задач были определены требования к ПС такие как, поддержка одновременной работы 5-10 человек и возможность запуска клиентской части приложения на разных операционных системах: Windows XP/7/8/10, Linux, macOS**.**

# Проектирование ПС

## Используемые технологии и средства разработки

Выбор технологий является очень важным этапом разработки приложения. Выделим основные факторы, повлиявшие на выбор технологий:

* разрабатываемое ПО рассчитано на использование с платформой семейства Windows, но следует иметь в виду возможность смены платформы на такие операционные системы как Linux, macOS);
* в качестве технологий следует максимально ориентироваться на технологии с открытым исходным кодом;
* имеющийся опыт.

Основываясь на этих факторах, целесообразно выбрать для разработки платформу Java. Разрабатываемому приложению потребуется выполнять массу разнообразных задач, для реализации которых крайне желательно использование сторонних библиотек. Платформа Java обеспечена поддержкой множества фреймворков, подходящих для реализации практически любой задачи. Кроме того, сам язык Java является элегантным, мультипарадигменным языком, позволяющим с легкостью писать красивый, поддерживаемый, расширяемый код.

### **Программная платформа Java**

Программная платформа Java - ряд продуктов и спецификаций компании Sun Microsystems, ныне являющейся дочерней компанией корпорации Oracle, которые предоставляют систему для разработки программного обеспечения и развертывания его на любой платформе. Java используется в совершенно разнообразных системах, начиная от настольных компьютеров, заканчивая встраиваемыми устройствами. Язык Java отличают следующие особенности:

- Простота и мощь. Одним из ключевых факторов философии Java является простота обучения и написания кода.

- Безопасность. Программы на Java запускаются изолированно и не получают напрямую доступ к системным ресурсам.

- Объектная ориентированность. Объектная модель представлена в множестве языков. Java придерживается строгих принципов ООП, максимально стараясь не нарушать его парадигмы, но в то же время легко расширяется.

- Надежность. Java является языком со строгой типизацией, что является в современном мире как плюсом, так и минусом. Однако, благодаря строгой типизации Java способствует обнаружению ошибок на ранних этапах разработки программ. Кроме того, обеспечивается динамическое управление памятью, что является большим плюсом, так как в платформах, не обладающих такой возможностью, одной из главных из сложных проблем являются труднообнаружимые и зачастую критические ошибки, связанные с утечками памяти.

- Мультиплатформенность. Разработчики Java позволили не на словах, а на деле, написав программу один раз, запускать ее на любой платформе, где существует виртуальная машина Java.

Виртуальная машина Java - основная часть Java Runtime Environment. Виртуальная машина исполняет байт-код Java, предварительно созданный компилятором из исходного кода. JVM может также использоваться для запуска программ на других языках, не только на Java. Обычно виртуальные машины Java содержат интерпретатор байт-кода и JIT-компилятор. Схема работы системы и набор байт-кодов виртуальной машины Java таковы, что позволяют достичь высокой производительности на этапе выполнения программы:

- код анализируется на соблюдение правил безопасности до запуска кода на выполнение

- работа с базовыми типами максимально эффективна

- методы в классах могут связываться как динамически, так и статически

- автоматический сборщик мусора работает в фоновом режиме, не замедляя основной поток исполнения и обеспечивает возврат свободной памяти в систему

- стандарт Java позволяет при необходимости писать критические к производительности участки кода на языках, являющимися нативными для целевой платформы

- язык Java обладает поддержкой многопоточности на уровне языка - часть примитивов синхронизации встроена в систему реального времени.

Важным фактором является и то, что Java включает в себя набор стандартных библиотек, которые можно разбить на следующие пакеты:

- java.lang - базовый набор типов

- java.io - потоки и файлы произвольного доступа

- java.util - коллекции и некоторые другие утилиты

### **NetBeans**

NetBeans IDE — свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, Python, PHP, JavaScript, C, C++, Ада[3] и ряда других.

Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведётся независимым сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org.

Последние версии NetBeans IDE поддерживают рефакторинг, профилирование, выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету и множество предопределённых шаблонов кода.

Для разработки программ в среде NetBeans и для успешной инсталляции и работы самой среды NetBeans должен быть предварительно установлен Sun JDK или J2EE SDK подходящей версии. Среда разработки NetBeans по-умолчанию поддерживала разработку для платформ J2SE и J2EE. Начиная с версии 6.0 NetBeans поддерживает разработку для мобильных платформ J2ME, C++ (только g++) и PHP без установки дополнительных компонентов.[4]

### **JavaFX**

JavaFX представляет инструментарий для создания кроссплатформенных графических приложений на платформе Java.

JavaFX позволяет создавать приложения с богатой насыщенной графикой благодаря использованию аппаратного ускорения графики и возможностей GPU.

С помощью JavaFX можно создавать программы для различных операционных систем: Windows, MacOS, Linux и для самых различных устройств: десктопы, смартфоны, планшеты, встроенные устройства, ТВ. Приложение на JavaFX будет работать везде, где установлена исполняемая среда Java (JRE).

JavaFX предоставляет большие возможности по сравнению с рядом других подобных платформ, в частности, по сравнению со Swing. Это и большой набор элементов управления, и возможности по работе с мультимедиа, двухмерной и трехмерной графикой, декларативный способ описания интерфейса с помощью языка разметки FXML, возможность стилизации интерфейса с помощью CSS, интеграция со Swing и многое другое.

История JavaFX фактически началась в первой половине 2000-х годов, когда разработчик по имени Крис Оливер (Chris Oliver), будучи работником компании SeeBeyond, разработал для создания графических интерфейсов новый язык F3 (Froms Follows Functions). Впоследствии в 2005 году SeeBeyond была приобретена компанией Sun Microsystems (которая на тот момент развивала язык Java до покупки компанией Oracle). F3 был переименован в JavaFX, а Крис Оливер продолжил работу над новой платформой уже в рамках компании Sun. И в мае 2007 года Sun Microsystems публично анонсировала новую платформу для создания графических приложений. А 4 декабря 2008 года вышел JavaFX 1.0 SDK.

После приобретения Sun Microsystems компанией Oracle в 2010 году была анонсирована, а в 2011 году вышла в релиз версия JavaFX 2.0. В первой версии JavaFX фактически представлял скиптовый язык. Во второй версии был полностью изменен подход. Скриптовый язык был убран, а платформа была полностью переписана фактически с нуля. Теперь создавать приложения можно было с помощью любого языка, который поддерживала JVM. Были добавлены новые API, интеграция со Swing и много других вещей.

Следующими важными вехами в развитии платформы стали версии JavaFX 8 и особенно JavaFX 9, которая вышла в сентябре 2017 года вместе с Java 9 и привнесла в платформу модульность.

В сентябре 2018 года вышла последняя версия фреймворка - JavaFX 11, которая была отделена от основной функциональности Java SE и стала использоваться как отдельный модуль.

На данный момент JavaFX представляет предпочтительный способ для создания графических приложений с помощью языка Java, который пришел на смену AWT и Swing. Также стоит отметить, что для работы с JavaFX вместо Java теоретически можно использовать любой язык программирования, который поддерживается JVM.[3]

### **MySQL**

MySQL – реляционная система управления базами данных со свободным кодом. Распространяется как свободно, так и под коммерческой лицензией. MySQL является решением для малых и средних приложений. СУБД портирована на большое количество платформ, имеет API для множества языков. MySQL была выбрана в качестве реляционной базы данных для проекта из-за того, что она максимально удовлетворяет потребностям: на БД не ожидается большой нагрузки, развернуть ее можно на большинстве популярных платформ. Для связи с БД используем библиотеку JOOQ. Это легковесная современная библиотека, написанная в функциональном стиле и идеально интегрируется с возможностями Java 8, в частности StreamAPI.[2]

### **Библиотека** **Apache POI**

Apache POI - предоставляет чистые библиотеки Java для чтения и записи файлов в форматах Microsoft Office, таких как Word, PowerPoint и Excel.

## Архитектура приложения

Проект представляет собой клиент-серверное приложение, использующее Фреймворк JavaFX. Отдельного описания заслуживает используемый при разработке шаблон MVC.

**Model-view-controller** (MVC, «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данная схема проектирования часто используется для построения архитектурного каркаса, когда переходят от теории к реализации в конкретной предметной области.



Концепция MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

* Модель (англ. Model). Модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать.
* Представление, вид (англ. View). Отвечает за отображение информации (визуализацию). Часто в качестве представления выступает форма (окно) с графическими элементами.
* Контроллер (англ. Controller). Обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

В ходе проектирования ПС были выбраны:

* язык программирования java обоснованный необходимостью кроссплатформенности ПС;
* среда разработки приложений NetBeans обоснованная своей доступностью;
* технология JavaFX позволяющая создавать приложения с богатой насыщенной графикой благодаря использованию аппаратного ускорения графики и возможностей GPU;
* чистая библиотека Java - Apache POI для формирования сводных ведосомтей материалов в формате данных Microsoft Excel;
* база данных MySQL обоснованная своей доступностью.

А также был выбран шаблон MVC основной целью применения концепции которого состоит в отделении бизнес логики(модели) от её визуализации (представления, вида).

# Разработка ПС

## Реализация клиентского-приложения

Перед вами дерево файлов/папок проекта:



Основные сущности реализованы в папке Models.

Сущность CommonMatreials описывает вспомогательные материаоы. Описана в файле CommonMatreials.java.

Сущность CommonMaterialsValues описывает параметры вспомогательных материалов. Описана в файле CommonMaterialsValues.java

Сущность Elements описывает элементы соответствующие обозначениям в конструкторской документации. Описана в Elements.java

Сущность Formulas описывает формулы материалов. Описана в Formulas.java

Сущность Materials описывает материалы. Описана в Materials.java

Сущность MaterialsValues описывает параметры материалов. Описана в MaterialsValues.java

Сущность Node описывает узел иерархической структуры данных. Описана в Node.java

Сущность Orders описывает заказы. Описана в Orders.java

Сущность Schedule предназначена для проставления даты в графике учёта нормированных узлов. Описана в Schedule.java

Сущность Specification описывает конструкторскую спецификацию. Описана в Specification.java

Сущность SpecificationCommonMaterials параметры вспомогательного материала. Описана в SpecificationCommonMaterials.java

Контролеры описаны в папке *Controllers*.

Контроллер *MainViewController* предоставляет взаимодействие с основными функциями приложения.

Контроллер *ParametersController* предоставляет диалоговое окно для введения параметров вспомогательного материала.

Контроллер *SplashScreenController* предоставляет загрузочный экран.

Представления находятся в папке *Views*. Для простоты отметим, что по соглашениям и для корректной работы, представления для каждого контроллера находится в соответствующей папке с тем же именем, что и имя контроллера.

В ходе разработки ПС было разработано согласно шаблону MVC: классы модели 11 шт., классы контроллеры и представления 3 шт. соответственно.

* 1. **Описание основных классов и их методов**

**Класс Desktop** позволяет взаимодействовать с различными возможностями рабочего стола.

Поддерживаемые операции включают в себя:

* запуск браузера по умолчанию для пользователя, чтобы показать указанный URI;
* запуск почтового клиента по умолчанию с дополнительным URI mailto;
* запуск зарегистрированного приложения для открытия, редактирования или печати указанного файла.

Этот класс предоставляет методы, соответствующие этим операциям. Методы ищут соответствующее приложение, зарегистрированное на текущей платформе, и запускают его для обработки URI или файла. Если связанного приложения нет или не удалось запустить связанное приложение, выдается исключение.

**Класс Alert** подклассирует класс Dialog и обеспечивает поддержку ряда предварительно созданных типов диалогов, которые могут быть легко показаны пользователям для запроса ответа. Поэтому для многих пользователей класс Alert является наиболее подходящим классом для их нужд (в отличие от непосредственного использования Dialog). В качестве альтернативы, пользователям, которые хотят предложить пользователю ввести текст или сделать выбор из списка параметров, будет лучше обслуживаться с использованием TextInputDialog и ChoiceDialog, соответственно.

При создании экземпляра Alert пользователи должны передать значение перечисления Alert.AlertType. Именно путем передачи этого значения экземпляр Alert будет настраиваться соответствующим образом (путем установки значений по умолчанию для многих свойств диалога, включая заголовок, заголовок и графику, а также кнопок по умолчанию, которые ожидаются в диалоговом окне данного тип.

Чтобы создать (но еще не показывать) предупреждение, просто используйте код, например, такой: Alert alert = new Alert (AlertType.CONFIRMATION, «Вы уверены, что хотите отформатировать свою систему?»);

После того, как Alert создан, мы должны показать его. Чаще всего оповещения (и диалоговые окна в целом) отображаются в модальном и блокирующем режиме. «Модальный» означает, что диалоговое окно предотвращает взаимодействие пользователя с приложением-владельцем, пока оно отображается, а «блокирование» означает, что выполнение кода останавливается в точке, в которой отображается диалоговое окно. Это означает, что вы можете отобразить диалоговое окно, дождаться ответа пользователя, а затем продолжить выполнение кода, который следует непосредственно за вызовом show, давая разработчикам возможность сразу же обрабатывать вводимые пользователем данные из диалога (при необходимости).

Диалоги JavaFX по умолчанию являются модальными (вы можете изменить это с помощью API Dialog.initModality (javafx.stage.Modality)). Чтобы указать, хотите ли вы блокировать или не блокировать диалоги, разработчики просто выбирают вызов Dialog.showAndWait () или Dialog.show () (соответственно). По умолчанию большинство разработчиков должны использовать Dialog.showAndWait (), учитывая простоту кодирования в этих ситуациях. Ниже показаны три фрагмента кода, показывающие три одинаково правильных способа отображения диалогового окна Alert, которое было указано выше.

**Класс File** - это представление Java файла или пути к каталогу. Поскольку имена файлов и каталогов имеют разные форматы на разных платформах, простая строка не подходит для их имен. Класс File содержит несколько методов для работы с именем пути, удаления и переименования файлов, создания новых каталогов, перечисления содержимого каталога и определения нескольких общих атрибутов файлов и каталогов.

Это абстрактное представление имен файлов и каталогов.

Имя пути, будь то абстрактное или в виде строки, может быть абсолютным или относительным. Родитель абстрактного пути может быть получен путем вызова метода getParent () этого класса.

Прежде всего, мы должны создать объект класса File, передав ему имя файла или имя каталога. Файловая система может реализовывать ограничения для определенных операций с фактическим объектом файловой системы, таких как чтение, запись и выполнение. Эти ограничения известны как разрешения на доступ.

Экземпляры класса File являются неизменяемыми; то есть после создания абстрактный путь, представленный объектом File, никогда не изменится.

**Класс FileOutputStream** принадлежит байтовому потоку и хранит данные в виде отдельных байтов. Может использоваться для создания текстовых файлов. Файл представляет собой хранилище данных на втором носителе, таком как жесткий диск или компакт-диск. Доступен ли файл или может быть создан, зависит от базовой платформы. В частности, некоторые платформы позволяют открывать файл для записи только одним FileOutputStream (или другими объектами записи файлов) за раз. В таких ситуациях конструкторы в этом классе потерпят неудачу, если соответствующий файл уже открыт.

FileOutputStream предназначен для записи потоков необработанных байтов, таких как данные изображения. Для записи потоков символов рассмотрите возможность использования FileWriter.

Важные методы:

void close (): закрывает поток вывода этого файла и освобождает все системные ресурсы, связанные с этим потоком.

protected void finalize (): очищает соединение с файлом и гарантирует, что метод close этого потока вывода файла вызывается, когда больше нет ссылок на этот поток.

void write (byte [] b): записывает байты b.length из указанного массива байтов в этот поток вывода файла.

void write (byte [] b, int off, int len): записывает len байтов из указанного массива байтов, начиная со смещения, в этот поток вывода файла.

void write (int b): записывает указанный байт в этот поток вывода файла.

**Класс Bufferreader** записывает текст в символьный поток вывода, буферизуя символы. Таким образом, обеспечивая эффективную запись одного массива, символа и строк. Размер буфера должен быть указан, если нет, он принимает значение по умолчанию.

Writer немедленно устанавливает выход в основной символ или поток байтов.

**Класс OutputStreamWriter** - это мост между символьными потоками и байтовыми потоками: записанные в него символы кодируются в байты с использованием указанной кодировки. Используемая им кодировка может быть указана по имени или задана явно, или может быть принят кодовый набор по умолчанию для платформы.

Каждый вызов метода write () приводит к тому, что преобразователь кодирования вызывается для данного символа (символов). Полученные байты накапливаются в буфере перед записью в основной выходной поток. Обратите внимание, что символы, передаваемые в методы write (), не буферизуются.

**Класс ArrayList** является частью инфраструктуры коллекции и присутствует в пакете java.util. Он предоставляет нам динамические массивы в Java. Хотя это может быть медленнее, чем стандартные массивы, но может быть полезно в программах, где требуется много манипуляций в массиве.

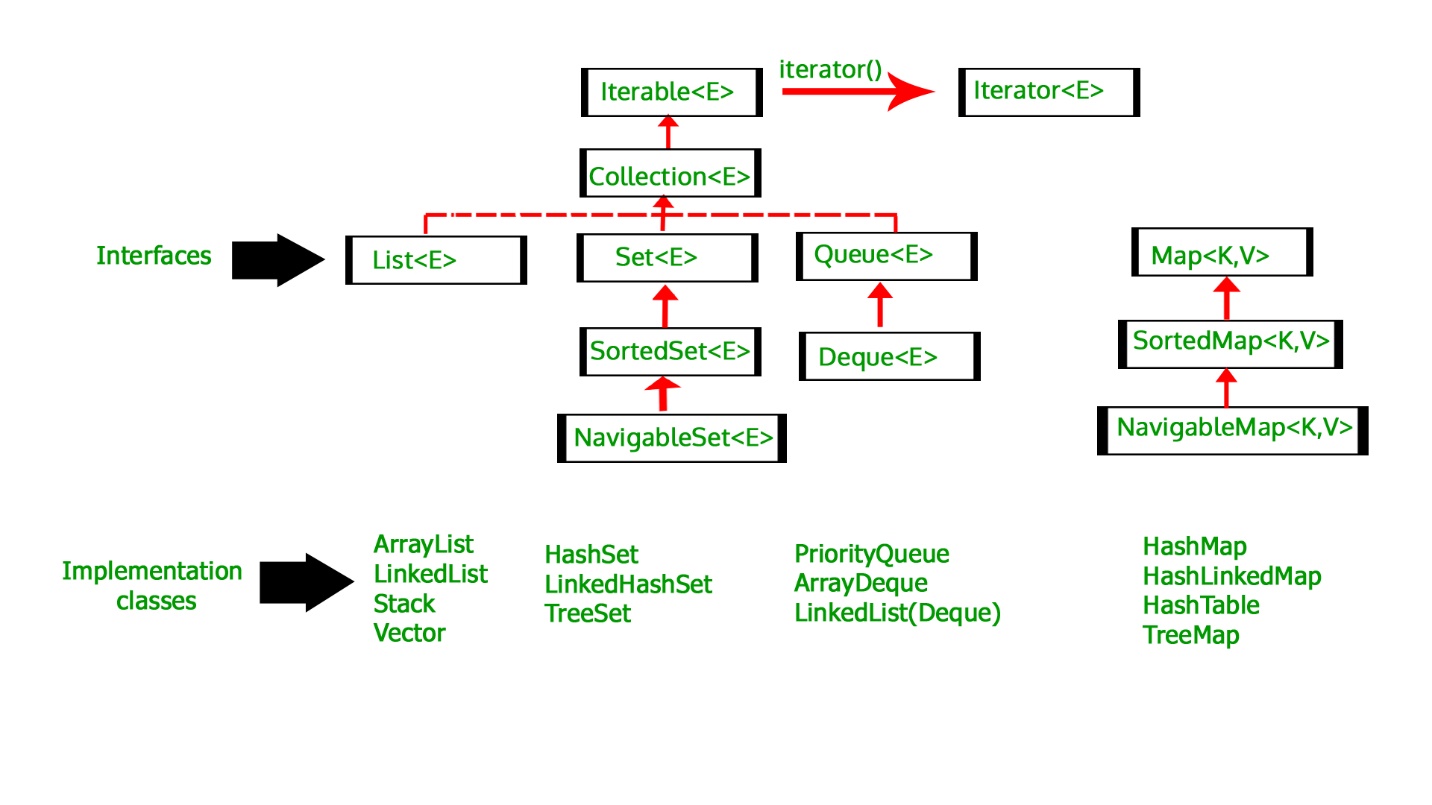
ArrayList наследует класс AbstractList и реализует интерфейс List.

ArrayList инициализируется размером, однако размер может увеличиваться, если коллекция увеличивается или уменьшается, если объекты удаляются из коллекции.

Java ArrayList позволяет нам случайный доступ к списку.

ArrayList не может использоваться для примитивных типов, таких как int, char и т. Д. Для таких случаев нам нужен класс-оболочка (подробности см. В этом разделе).

ArrayList в Java можно рассматривать как схожий с вектором в C ++.



**Класс Date** представляет определенный момент времени с точностью до миллисекунды.

До JDK 1.1 класс Date имел две дополнительные функции. Это позволило интерпретировать даты как значения года, месяца, дня, часа, минуты и секунды. Это также позволило форматировать и анализировать строки даты. К сожалению, API для этих функций не поддается интернационализации. Начиная с JDK 1.1, класс Calendar должен использоваться для преобразования полей даты и времени, а класс DateFormat должен использоваться для форматирования и анализа строк даты. Соответствующие методы в Date устарели.

Хотя класс Date предназначен для отражения координированного универсального времени (UTC), он может делать это не совсем точно, в зависимости от среды хоста виртуальной машины Java. Почти все современные операционные системы предполагают, что 1 день = 24 × 60 × 60 = 86400 секунд во всех случаях. В UTC, однако, примерно раз в год или два появляется дополнительная секунда, называемая «високосной секундой». Дополнительная секунда дня всегда добавляется как последняя секунда дня и всегда 31 декабря или 30 июня. Например, последняя минута 1995 года была длиной 61 секунда, благодаря добавленной дополнительной секунде. Большинство компьютерных часов недостаточно точны, чтобы отражать разницу в секунду.

Некоторые компьютерные стандарты определены в терминах среднего времени по Гринвичу (GMT), которое эквивалентно универсальному времени (UT). GMT - «гражданское» название стандарта; UT - это «научное» название для того же стандарта. Различие между UTC и UT состоит в том, что UTC основан на атомных часах, а UT основан на астрономических наблюдениях, которые для всех практических целей представляют собой незаметно тонкие волосы для расщепления. Поскольку вращение Земли не является равномерным (оно замедляется и ускоряется сложными способами), UT не всегда течет равномерно. По мере необходимости в UTC вводятся дополнительные секунды, чтобы UTC оставалось в пределах 0,9 секунды от UT1, который является версией UT с некоторыми внесенными исправлениями. Есть и другие системы времени и даты; например, шкала времени, используемая спутниковой системой глобального позиционирования (GPS), синхронизируется с UTC, но не регулируется для високосных секунд. Интересным источником дополнительной информации является Военно-морская обсерватория США (USNO):

http://www.usno.navy.mil/USNO

и материал о «Системах времени» по адресу:

http://www.usno.navy.mil/USNO/time/master-clock/systems-of-time

который имеет описания различных систем времени, включая UT, UT1 и UTC.

Во всех методах класса Date, которые принимают или возвращают значения года, месяца, даты, часов, минут и секунд, используются следующие представления:

Год y представлен целым числом y - 1900.

Месяц представлен целым числом от 0 до 11; 0 - январь, 1 - февраль и т. Д .; таким образом, 11 декабря.

Дата (день месяца) представляется целым числом от 1 до 31 в обычном порядке.

Час представлен целым числом от 0 до 23. Таким образом, час с полуночи до 1 часа ночи - это час 0, а час с полудня до 1 часа дня. Час 12

Минута представляет собой целое число от 0 до 59 в обычном порядке.

Секунду представляет целое число от 0 до 61; значения 60 и 61 встречаются только в течение високосных секунд и даже тогда только в тех реализациях Java, которые фактически правильно отслеживают високосные секунды. Из-за способа, которым в настоящее время вводятся високосные секунды, крайне маловероятно, что две високосные секунды будут происходить в одну и ту же минуту, но эта спецификация соответствует соглашениям о дате и времени для ISO C.

Во всех случаях аргументы, данные методам для этих целей, не обязательно должны попадать в указанные диапазоны; например, дата может быть указана как 32 января и интерпретируется как означающая 1 февраля.

**Класс SimpleDateFormat** - это конкретный класс для форматирования и анализа дат с учетом языка. Он позволяет форматировать (дата -> текст), анализировать (текст -> дата) и нормализовать.

SimpleDateFormat позволяет начать с выбора любых пользовательских шаблонов для форматирования даты и времени. Тем не менее, рекомендуется создать форматер даты и времени с помощью getTimeInstance, getDateInstance или getDateTimeInstance в DateFormat. Каждый из этих методов класса может возвращать форматер даты / времени, инициализированный шаблоном формата по умолчанию. При желании вы можете изменить шаблон формата, используя методы applyPattern. Для получения дополнительной информации об использовании этих методов см. DateFormat.

Шаблоны даты и времени

Форматы даты и времени указываются в шаблонах даты и времени. В строках шаблонов даты и времени буквы без кавычек от «A» до «Z» и от «a» до «z» интерпретируются как буквы шаблонов, представляющие компоненты строки даты или времени. Текст может быть заключен в кавычки ('), чтобы избежать интерпретации. «» обозначает одну кавычку. Все остальные символы не интерпретируются; они просто копируются в выходную строку во время форматирования или сопоставляются с входной строкой во время синтаксического анализа.

Определены следующие символы шаблона (все остальные символы от 'A' до 'Z' и от 'a' до 'z' зарезервированы):

Таблица 4.2.1 – Символы

|  |  |  |  |
| --- | --- | --- | --- |
| **Символ** | **Дата или время компанента** | **Вывод** | **Образец** |
| G | Era designator | [Text](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#text) | AD |
| y | Year | [Year](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#year) | 1996; 96 |
| Y | Week year | [Year](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#year) | 2009; 09 |
| M | Month in year | [Month](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#month) | July; Jul; 07 |
| w | Week in year | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 27 |
| W | Week in month | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 2 |
| D | Day in year | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 189 |
| d | Day in month | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 10 |
| F | Day of week in month | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 2 |
| E | Day name in week | [Text](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#text) | Tuesday; Tue |
| u | Day number of week (1 = Monday, ..., 7 = Sunday) | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 1 |
| a | Am/pm marker | [Text](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#text) | PM |
| H | Hour in day (0-23) | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 0 |
| k | Hour in day (1-24) | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 24 |
| K | Hour in am/pm (0-11) | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 0 |
| h | Hour in am/pm (1-12) | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 12 |
| m | Minute in hour | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 30 |
| s | Second in minute | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 55 |
| S | Millisecond | [Number](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#number) | 978 |
| z | Time zone | [General time zone](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#timezone) | Pacific Standard Time; PST; GMT-08:00 |
| Z | Time zone | [RFC 822 time zone](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#rfc822timezone) | -0800 |
| X | Time zone | [ISO 8601 time zone](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html#iso8601timezone) | -08; -0800; -08:00 |

Шаблон букв обычно повторяется, так как их количество определяет точное представление:

**Text**: для форматирования, если количество букв шаблона составляет 4 или более, используется полная форма; в противном случае используется короткая или сокращенная форма, если таковая имеется. Для синтаксического анализа принимаются обе формы, независимо от количества букв шаблона.

**Number**: Для форматирования количество букв шаблона - это минимальное количество цифр, а более короткие числа дополняются нулями до этой суммы. При синтаксическом анализе количество букв шаблона игнорируется, если только это не необходимо для разделения двух смежных полей.

**Year**: если календарь форматера является григорианским, применяются следующие правила.

Для форматирования, если количество букв шаблона равно 2, год усекается до 2 цифр; в противном случае он интерпретируется как число.

Для синтаксического анализа, если количество букв шаблона превышает 2, год интерпретируется буквально, независимо от количества цифр. Таким образом, используя шаблон «MM / dd / yyyy», «01/11/12» анализирует до 11 января 12 г. н.э.

Для анализа с сокращенным шаблоном года ("y" или "yy") SimpleDateFormat должен интерпретировать сокращенный год относительно некоторого столетия. Это достигается путем корректировки дат в пределах 80 лет до и 20 лет после времени создания экземпляра SimpleDateFormat. Например, используя шаблон «MM / dd / yy» и экземпляр SimpleDateFormat, созданный 1 января 1997 года, строка «01/11/12» будет интерпретироваться как 11 января 2012 года, а строка «05/04 / 64 "будет интерпретироваться как 4 мая 1964 года. Во время синтаксического анализа только строки, состоящие ровно из двух цифр, как определено Character.isDigit (char), будут проанализированы в столетии по умолчанию. Любая другая числовая строка, такая как строка из одной цифры, строка из трех или более цифр или строка из двух цифр, которая не содержит все цифры (например, «-1»), интерпретируется буквально. Таким образом, "01/02/3" или "01/02/003" анализируются, используя тот же шаблон, что и 2 января 3 года нашей эры. Аналогично, "01/02 / -3" анализируется как 2 января 4 года до нашей эры.

В противном случае применяются специальные формы календарной системы. Как для форматирования, так и для разбора, если число букв шаблона составляет 4 или более, используется специальная длинная форма календаря. В противном случае используется специальная краткая или сокращенная форма календаря.

Если указан год недели «Y» и календарь не поддерживает ни одного года недели, вместо него используется календарный год («y»). Поддержка дней недели может быть проверена с помощью вызова getCalendar (). IsWeekDateSupported ().

**Month**: если количество букв шаблона 3 или более, месяц интерпретируется как текст; в противном случае он интерпретируется как число.

**General time zone**: Часовые пояса интерпретируются как текст, если они имеют имена. Для часовых поясов, представляющих значение смещения по Гринвичу, используется следующий синтаксис:

GMTOffsetTimeZone:

GMT Sign Hours : Minutes

Sign: one of

+ -

Hours:

Digit

Digit Digit

Minutes:

Digit Digit

Digit: one of

0 1 2 3 4 5 6 7 8 9

Часы должны быть от 0 до 23, а минуты - от 00 до 59. Формат не зависит от локали, а цифры должны быть взяты из блока Basic Latin в стандарте Unicode.

Для анализа также принимаются часовые пояса RFC 822.

**RFC 822 time zone**: Для форматирования используется 4-значный формат часового пояса RFC 822:

RFC822TimeZone:

Sign TwoDigitHours Minutes

TwoDigitHours:

Digit Digit

TwoDigitHours должно быть между 00 и 23. Другие определения такие же, как для общих часовых поясов.

Для разбора также принимаются общие часовые пояса.

**ISO 8601 Time zone**. Количество букв шаблона обозначает формат для форматирования и синтаксического анализа следующим образом:

ISO8601TimeZone:

OneLetterISO8601TimeZone

TwoLetterISO8601TimeZone

ThreeLetterISO8601TimeZone

OneLetterISO8601TimeZone:

Sign TwoDigitHours

Z

TwoLetterISO8601TimeZone:

Sign TwoDigitHours Minutes

Z

ThreeLetterISO8601TimeZone:

Sign TwoDigitHours : Minutes

Z

Другие определения как для **General time zone** или **RFC 822 time zone**.

Для форматирования, если значение смещения от GMT равно 0, создается «Z». Если количество букв шаблона равно 1, любая доля часа игнорируется. Например, если шаблон «X», а часовой пояс «GMT + 05: 30», создается «+05».

Для анализа "Z" анализируется как указатель часового пояса UTC. Общие часовые пояса не принимаются.

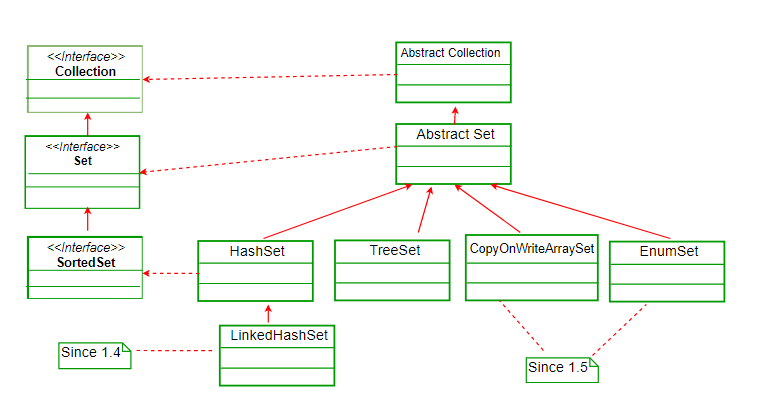
Если количество букв шаблона составляет 4 или более, IllegalArgumentException генерируется при создании SimpleDateFormat или применении шаблона.

SimpleDateFormat также поддерживает локализованные строки шаблонов даты и времени. В этих строках буквы шаблона, описанные выше, могут быть заменены другими, зависящими от локали, шаблонными буквами. SimpleDateFormat не имеет дело с локализацией текста, кроме букв шаблона; это зависит от клиента класса.

**Класс HashSet** реализует интерфейс Set, поддерживаемый хеш-таблицей, которая на самом деле является экземпляром HashMap. Не дается никаких гарантий относительно порядка итераций набора, что означает, что класс не гарантирует постоянный порядок элементов во времени. Этот класс допускает нулевой элемент. Класс также обеспечивает постоянную производительность по времени для основных операций, таких как добавление, удаление, удержание и размер, при условии, что хеш-функция правильно распределяет элементы между сегментами, что мы увидим далее в статье.

Несколько важных функций HashSet:

* Реализует Set Interface.
* Базовая структура данных для HashSet является хеш-таблицей.
* Поскольку он реализует интерфейс Set, повторяющиеся значения не допускаются.
* Объекты, которые вы вставляете в HashSet, не обязательно будут вставлены в том же порядке. Объекты вставляются на основе их хэш-кода.
* Элементы NULL разрешены в HashSet.
* HashSet также реализует интерфейсы Searlizable и Cloneable.



**Initial Capacity(Начальная емкость)**: Начальная емкость означает число сегментов, когда создается хеш-таблица (HashSet внутренне использует структуру данных хеш-таблицы). Количество сегментов будет автоматически увеличено, если текущий размер заполнится.

**Load Factor(Коэффициент загрузки)**: Коэффициент загрузки является мерой того, насколько полно HashSet может получить до того, как его емкость будет автоматически увеличена. Когда количество записей в хеш-таблице превышает произведение коэффициента загрузки и текущей емкости, хеш-таблица перефразируется (то есть внутренние структуры данных перестраиваются), так что хеш-таблица имеет приблизительно вдвое больше сегментов.

Number of stored elements in the table

load factor = -----------------------------------------

                        Size of the hash table

**Класс Optional** это контейнерный объект, который может содержать или не содержать ненулевое значение. Если значение присутствует, isPresent () возвращает true, а get () возвращает значение.

Предоставляются дополнительные методы, которые зависят от наличия или отсутствия содержащегося в нем значения, например orElse () (возвращает значение по умолчанию, если значение отсутствует) и ifPresent () (выполняет действие, если значение присутствует).

Это класс на основе значений; использование чувствительных к идентичности операций (включая равенство ссылок (==), хэш-код идентичности или синхронизацию) в экземплярах Optional может привести к непредсказуемым результатам, и его следует избегать.

**Интерфейс Set<E>** – это коллекция, которая не содержит повторяющихся элементов. Более формально, множества не содержат пары элементов e1 и e2, таких как e1.equals (e2), и не более одного нулевого элемента. Как следует из его названия, этот интерфейс моделирует абстракцию математического набора.

Интерфейс Set помещает дополнительные условия, помимо тех, которые унаследованы от интерфейса Collection, в контракты всех конструкторов и в контракты методов add, equals и hashCode. Объявления для других унаследованных методов также включены сюда для удобства. (Спецификации, сопровождающие эти объявления, были адаптированы к интерфейсу Set, но они не содержат каких-либо дополнительных условий.)

Дополнительным условием для конструкторов, что неудивительно, является то, что все конструкторы должны создавать набор, который не содержит повторяющихся элементов (как определено выше).

Примечание. Необходимо соблюдать осторожность, если в качестве элементов набора используются изменяемые объекты. Поведение набора не указывается, если значение объекта изменяется таким образом, что это влияет на сравнение равных, в то время как объект является элементом в наборе. Особый случай этого запрета состоит в том, что недопустимо, чтобы набор содержал себя как элемент.

Некоторые реализации множеств имеют ограничения на элементы, которые они могут содержать. Например, некоторые реализации запрещают нулевые элементы, а некоторые имеют ограничения на типы их элементов. Попытка добавить неприемлемый элемент вызывает неконтролируемое исключение, обычно NullPointerException или ClassCastException. Попытка запросить наличие неприемлемого элемента может вызвать исключение или просто вернуть false; некоторые реализации будут демонстрировать первое поведение, а некоторые - второе. В более общем смысле, попытка выполнить операцию с неподходящим элементом, завершение которого не приведет к вставке неприемлемого элемента в набор, может вызвать исключение или может быть успешным, по выбору реализации. Такие исключения помечены как «необязательные» в спецификации для этого интерфейса.

Статические фабричные методы неизменного набора

Статические фабричные методы Set.of () предоставляют удобный способ создания неизменяемых наборов. Экземпляры Set, созданные этими методами, имеют следующие характеристики:

* Они структурно неизменны. Элементы не могут быть добавлены или удалены. Вызов любого метода-мутатора всегда вызывает исключение UnsupportedOperationException. Однако, если содержащиеся элементы сами являются изменяемыми, это может привести к тому, что Set будет вести себя непоследовательно или его содержимое может измениться.
* Они запрещают нулевые элементы. Попытки создать их с нулевыми элементами приводят к исключению NullPointerException.
* Они сериализуемы, если все элементы сериализуемы.
* Они отклоняют дубликаты элементов во время создания. Повторяющиеся элементы, передаваемые статическому фабричному методу, приводят к IllegalArgumentException.
* Порядок итерации заданных элементов не определен и может быть изменен.
* Они основаны на стоимости. Вызывающие абоненты не должны делать никаких предположений относительно личности возвращенных экземпляров. Фабрики могут создавать новые экземпляры или повторно использовать существующие. Поэтому чувствительные к идентичности операции в этих случаях (ссылочное равенство (==), хэш-код идентичности и синхронизация) ненадежны и их следует избегать.
* Они сериализуются, как указано на странице «Сериализованная форма».

Этот интерфейс является членом Java Collections Framework.

**Класс TreeSet** является одной из наиболее важных реализаций интерфейса SortedSet в Java, который использует дерево для хранения. Порядок элементов поддерживается набором с использованием их естественного порядка, независимо от того, предоставляется ли явный компаратор. Это должно быть согласовано с equals, если необходимо правильно реализовать интерфейс Set. Он также может быть заказан компаратором, предоставляемым во время создания набора, в зависимости от того, какой конструктор используется. TreeSet реализует интерфейс NavigableSet, наследуя класс AbstractSet.

Несколько важных функций TreeSet:

* TreeSet реализует интерфейс SortedSet, поэтому повторяющиеся значения не допускаются.
* Объекты в TreeSet хранятся в отсортированном и возрастающем порядке.
* TreeSet не сохраняет порядок вставки элементов, но элементы сортируются по ключам.
* TreeSet не позволяет вставлять гетерогенные объекты. Он выдаст исключение classCastException во время выполнения, если попытается добавить гетерогенные объекты.
* TreeSet служит отличным выбором для хранения больших объемов отсортированной информации, к которой предполагается быстрый доступ из-за более быстрого доступа и времени поиска.
* TreeSet - это, по сути, реализация самобалансирующегося бинарного дерева поиска, такого как Red-Black Tree. Поэтому такие операции, как добавление, удаление и поиск, занимают O (Log n) время. А такие операции, как печать n элементов в отсортированном порядке, занимают O (n) времени.

Конструкторы класса TreeSet:

1. TreeSet t = new TreeSet ();

Это создаст пустой объект TreeSet, в котором элементы будут сохранены в порядке естественной сортировки по умолчанию.

1. TreeSet t = новый TreeSet (Comparator comp);

Этот конструктор используется, когда требуется внешняя спецификация порядка сортировки элементов.

1. TreeSet t = новый TreeSet (Коллекция col);

Этот конструктор используется, когда требуется любое преобразование из любого объекта Collection в объект TreeSet.

1. TreeSet t = новый TreeSet (SortedSet s);

Этот конструктор используется для преобразования объекта SortedSet в объект TreeSet.

Синхронизированный TreeSet:

Реализация в TreeSet не синхронизирована в том смысле, что если несколько потоков одновременно получают доступ к набору деревьев, и хотя бы один из потоков изменяет набор, он должен быть синхронизирован извне. Обычно это достигается путем синхронизации с некоторым объектом, который естественным образом инкапсулирует набор. Если такого объекта не существует, набор следует «обернуть» с помощью метода Collections.synchronizedSortedSet. Это лучше всего делать во время создания, чтобы предотвратить случайный несинхронизированный доступ к набору:

TreeSet ts = new TreeSet();

Set syncSet = Collections.synchronziedSet(ts);

**Класс Image** представляет графические изображения и используется для загрузки изображений с указанного URL-адреса.

Поддерживаемые форматы изображений:

* BMP
* GIF
* JPEG
* PNG

Размер изображения можно изменять по мере его загрузки (например, чтобы уменьшить объем памяти, потребляемой изображением). Приложение может указать качество фильтрации, используемое при масштабировании, а также указывать, следует ли сохранять соотношение сторон исходного изображения.

Все URL, поддерживаемые URL, могут быть переданы в конструктор. Если переданная строка не является допустимым URL-адресом, а является путем, вместо этого в этом случае выполняется поиск по образу в classpath.

Используйте ImageView для отображения изображений, загруженных с этим классом. Один и тот же экземпляр Image может отображаться несколькими ImageViews.

**Класс ImageView** - это узел, используемый для рисования изображений, загруженных с помощью класса Image.

Этот класс позволяет изменить размер отображаемого изображения (с сохранением или без сохранения исходного соотношения сторон) и указать область просмотра в исходном изображении для ограничения пикселей, отображаемых этим ImageView.

**Интерфейс ScriptEngine** - это фундаментальный интерфейс, методы которого должны быть полностью функциональными в каждой реализации этой спецификации.

Эти методы обеспечивают базовую функциональность сценариев. Ожидается, что приложения, написанные для этого простого интерфейса, будут работать с минимальными изменениями в каждой реализации. Он включает методы, которые выполняют сценарии, и те, которые устанавливают и получают значения.

Значения являются парами ключ / значение двух типов. Первый тип пар состоит из пар, ключи которых зарезервированы и определены в данной спецификации или в отдельных реализациях. Значения в парах с зарезервированными ключами имеют определенные значения.

Другой тип пар состоит из тех, которые создают привязки языка Java, значения обычно представлены в сценариях соответствующими ключами или их оформленными формами.

**Класс ScriptEngineManager** реализует механизм обнаружения и создания экземпляров для классов ScriptEngine, а также поддерживает коллекцию пар ключ / значение, хранящих состояние, совместно используемое всеми механизмами, созданными Manager. Этот класс использует механизм поставщика услуг для перечисления всех реализаций ScriptEngineFactory.

ScriptEngineManager предоставляет метод для возврата списка всех этих фабрик, а также служебные методы, которые ищут фабрики на основе имени языка, расширения файла и типа mime.

Привязки пар ключ / значение, именуемые «Глобальная область», поддерживаемые менеджером, доступны для всех экземпляров ScriptEngine, созданных ScriptEngineManager. Значения в привязках, как правило, отображаются во всех сценариях.

**Класс Platform** поддержки платформы приложения.

**Интерфейс Runnable** должен быть реализован любым классом, экземпляры которого предназначены для выполнения потоком. Класс должен определять метод без аргументов, который называется run.

Этот интерфейс предназначен для предоставления общего протокола для объектов, которые хотят выполнять код, пока они активны. Например, Runnable реализуется классом Thread. Быть активным просто означает, что поток был запущен и еще не был остановлен.

Кроме того, Runnable предоставляет средства для того, чтобы класс был активным, не наследуя Thread. Класс, реализующий Runnable, может выполняться без создания подкласса Thread, создавая экземпляр экземпляра Thread и передавая себя в качестве цели. В большинстве случаев интерфейс Runnable следует использовать, если вы планируете переопределить только метод run (), а не другие методы Thread. Это важно, потому что классы не должны быть разделены на подклассы, если программист не намеревается изменить или улучшить фундаментальное поведение класса.

**Метод public static void runLater​(Runnable runnable) класса Platform** запускает указанный Runnable в потоке приложений JavaFX в неустановленное время в будущем. Этот метод, который может быть вызван из любого потока, отправит Runnable в очередь событий, а затем сразу же вернется к вызывающей стороне. Runnables выполняются в порядке их размещения. Runnable, переданный в метод runLater, будет выполнен до того, как любой Runnable будет передан в последующий вызов runLater. Если этот метод вызывается после завершения работы JavaFX, вызов будет проигнорирован: Runnable не будет выполнен, и не будет выдано исключение.

ПРИМЕЧАНИЕ: приложения должны избегать переполнения JavaFX слишком большим количеством ожидающих Runnables. В противном случае приложение может перестать отвечать на запросы. Приложениям рекомендуется объединять несколько операций в меньшее количество вызовов runLater. Кроме того, по возможности следует выполнять длительные операции в фоновом потоке, освобождая поток приложений JavaFX для операций с графическим интерфейсом.

Этот метод нельзя вызывать до инициализации среды исполнения FX. Для стандартных приложений JavaFX, которые расширяют Application и используют для запуска приложения либо средство запуска Java, либо один из методов запуска в классе Application, среда запуска FX инициализируется модулем запуска перед загрузкой класса Application. Для приложений Swing, использующих JFXPanel для отображения содержимого FX, среда выполнения FX инициализируется при создании первого экземпляра JFXPanel. Для приложения SWT, использующего FXCanvas для отображения содержимого FX, среда выполнения FX инициализируется при создании первого экземпляра FXCanvas. Для приложений, которые не следуют ни одному из этих подходов, необходимо вручную запустить среду выполнения JavaFX, вызвав один раз запуск (Runnable).

**Класс DriverManager** предоставляет базовый сервис для управления набором драйверов JDBC.

В рамках своей инициализации класс DriverManager попытается загрузить классы драйверов, указанные в системном свойстве "jdbc.drivers". Это позволяет пользователю настраивать драйверы JDBC, используемые их приложениями. Например, в файле ~ / .hotjava / properties вы можете указать: jdbc.drivers = foo.bah.Driver: wombat.sql.Driver: bad.taste.ourDriver Программа также может явно загружать драйверы JDBC в любое время. Например, my.sql.Driver загружается с помощью следующего оператора: Class.forName ("my.sql.Driver");

Когда вызывается getConnection, DriverManager попытается найти подходящий драйвер среди тех, которые загружены при инициализации и явно загруженных с использованием того же загрузчика классов, что и текущий апплет или приложение.

**Интерфейсы JDBC Statement, CallableStatement и PreparedStatement** определяют методы и свойства, которые позволяют отправлять команды SQL или PL / SQL и получать данные из вашей базы данных.

Они также определяют методы, которые помогают преодолеть различия типов данных между типами данных Java и SQL, используемыми в базе данных.

В следующей таблице приведены сводные данные о назначении каждого интерфейса для выбора используемого интерфейса.

Таблица 4.2.2 – Интерфейсы

|  |  |
| --- | --- |
| **Интерфейсы** | **Рекомендуемое использование** |
| Statement | Заявление Используйте это для общего доступа к вашей базе данных. Полезно, когда вы используете статические операторы SQL во время выполнения. Интерфейс оператора не может принимать параметры. |
| PreparedStatement | Используйте это, когда вы планируете использовать операторы SQL много раз. Интерфейс PreparedStatement принимает входные параметры во время выполнения. |
| CallableStatement | Используйте это, если вы хотите получить доступ к хранимым процедурам базы данных. Интерфейс CallableStatement также может принимать входные параметры времени выполнения. |

**Коллекции в JavaFX**

Коллекция, иногда называемая контейнером, - это просто объект, который группирует несколько элементов в одну единицу. Коллекции используются для хранения, извлечения, обработки и передачи сводных данных. Как правило, они представляют элементы данных, которые образуют естественную группу, такие как покерная комбинация (набор карт), почтовая папка (набор букв) или телефонный справочник (сопоставление имен с телефонными номерами). Если вы использовали язык программирования Java - или почти любой другой язык программирования - вы уже знакомы с коллекциями.

Каркас коллекций - это унифицированная архитектура для представления коллекций и управления ими. Все рамки коллекций содержат следующее:

**Interfaces**: это абстрактные типы данных, которые представляют коллекции. Интерфейсы позволяют манипулировать коллекциями независимо от деталей их представления. В объектно-ориентированных языках интерфейсы обычно образуют иерархию.

**Implementations**: Это конкретные реализации интерфейсов коллекции. По сути, они являются структурами данных многократного использования.

**Algorithms**. Это методы, которые выполняют полезные вычисления, такие как поиск и сортировка, для объектов, которые реализуют интерфейсы коллекции. Говорят, что алгоритмы полиморфны: один и тот же метод может использоваться во многих различных реализациях соответствующего интерфейса коллекции. По сути, алгоритмы являются функциональностью многократного использования.

Коллекции в JavaFX определяются пакетом javafx.collections, который состоит из следующих интерфейсов и классов:

Таблица 4.2.3 – Интерфейсы

|  |  |
| --- | --- |
| **Интерфейсы** | **Описание** |
| ObservableList | Список, который позволяет нам отслеживать изменения |
| ListChangeListener | Интерфейс, который получает уведомления об изменениях |
| ObservableMap | Карта, которая позволяет нам отслеживать изменения |
| MapChangeListener | Интерфейс, который получает уведомления об изменениях из ObservableMap |

Таблица 4.2.4 – Классы

|  |  |
| --- | --- |
| **Классы** | **Описание** |
| FXCollections | Класс утилит отображается на java.util.Collections |
| ListChangeListener.Change | Represents a change made to an ObservableList |
| MapChangeListener.Change | Represents a change made to an ObservableMap |

Платформа коллекций Java предоставляет следующие преимущества:

* Сокращает усилия по программированию. Предоставляя полезные структуры данных и алгоритмы, Framework Коллекций освобождает вас от необходимости концентрироваться на важных частях вашей программы, а не на низкоуровневом «слесарном деле», необходимом для ее работы. Облегчая взаимодействие между несвязанными API, Java Collections Framework освобождает вас от написания объектов адаптера или кода преобразования для подключения API.
* Увеличивает скорость и качество программы: эта платформа коллекций обеспечивает высокопроизводительные, высококачественные реализации полезных структур данных и алгоритмов. Различные реализации каждого интерфейса являются взаимозаменяемыми, поэтому программы могут быть легко настроены путем переключения реализаций набора. Поскольку вы избавлены от тяжелой работы по написанию собственных структур данных, у вас будет больше времени для улучшения качества и производительности программ.
* Обеспечивает совместимость между несвязанными API: интерфейсы коллекций являются общеупотребительными, с помощью которых API передают коллекции вперед и назад. Если мой API сетевого администрирования предоставляет набор имен узлов и если ваш инструментарий GUI ожидает набор заголовков столбцов, наши API будут беспрепятственно взаимодействовать, даже если они были написаны независимо.
* Сокращает усилия по изучению и использованию новых API: многие API, естественно, принимают коллекции при вводе и предоставляют их в качестве вывода. В прошлом каждый такой API имел небольшой подчиненный API, предназначенный для управления своими коллекциями. Было мало последовательности между этими дополнительными API-интерфейсами коллекций, поэтому вам приходилось изучать каждый из них с нуля, и было легко допустить ошибки при их использовании. С появлением стандартных коллекционных интерфейсов проблема исчезла.
* Сокращает усилия по разработке новых API: это обратная сторона предыдущего преимущества. Дизайнеры и разработчики не должны заново изобретать колесо каждый раз, когда создают API, основанный на коллекциях; вместо этого они могут использовать стандартные интерфейсы коллекции.
* Поддерживает повторное использование программного обеспечения. Новые структуры данных, которые соответствуют стандартным интерфейсам сбора, по своей природе можно использовать повторно. То же самое касается новых алгоритмов, которые работают с объектами, которые реализуют эти интерфейсы.

**Класс FXCollection** – это служебный класс, состоящий из статических методов, которые представляют собой копии 1: 1 методов java.util.Collections.

Методы-обертки (такие как synchronizedObservableList или emptyObservableList) имеют точно такую ​​же функциональность, что и методы в коллекциях, за исключением того, что они возвращают ObservableList и поэтому подходят для методов, которые требуют ObservableList при вводе.

Служебные методы здесь в основном по соображениям производительности. Все методы оптимизированы таким образом, что они дают только ограниченное количество уведомлений. С другой стороны, методы java.util.Collections могут вызывать «методы модификации» в ObservableList несколько раз, что приводит к ряду уведомлений.

# Описание использования

## Система нормирования материалов в конструкторской документации

Главная окно программы выглядит следующим образом. С левой стороны располагаться следующие вкладки:

* Спецификация в неё вводятся все спецификации конструкторской документации.
* Дерево. Предназначена для просмотра/редактирования введённого узла в виде древовидного представления
* График. Предназначена для введения учёта пронумерованных узлов изделия.
* Заказы. Предназначена для добавления/удаления производственных заказов изделий.
* Обозначение. Предназначена для добавления/удаления/редактирования элементов(обозначений).
* Вспомогательные. Предназначена для добавления/удаления/редактирования вспомогательных материалов
* Формулы. Предназначена для добавления/удаления/редактирования формул материалов
* Материалы. Предназначена для добавления/удаления/редактирования материалов.
* С правой стороны располагаться 3 вкладки:
* Обозначение. Предназначена для редактирования элемента (обозначения) таких параметров как: Позиция, Обозначение, Кол(количество), ЕИ(единица измерения), Наименование, Примечание соответствующая конструкторской документации.
* Материал. Предназначена для присвоения материала деталям.
* Вспомогательные. Предназначена для присвоения вспомогательных материалов.



Для начало работы, следует нажать на кнопку D:\git\practise\img\home.bmp , которая перекинет в кореньевой каталог



Далее вводим обозначение по чертежу конструкторской документации ЛУИФ.К501-28.01.200. Нажимаем Enter или кнопку Добавить. В результате чего обозначение добавляется в конец списка.



Выбираем введенное обозначение и нажимаем кнопку D:\git\practise\img\right.bmp.



Далее вводим обозначения входящие в конструкторскую спецификацию



После добавления двух обозначений, проставляем их наименование и количество.





Теперь выбрав обозначение, переходим на вкладку материал, где выбираем материал соответствующий материалу на чертеже, вводим его параметры (длину, ширину) и проставляем участок на который будет далее получен материал. Проделываем аналогичное со вторым обозначением.



После завершения нормирования узла нажимаем кнопку D:\git\practise\img\left.bmp. Попадаем по иерархии на уровень выше. Выбрав пронумерованный узел ЛУИФ.К501-28.01.200 нажимаем кнопку файл->Сохранить НРМ.



Далее создается отчет в файле ЛУИФ.К501-28.01.200.xlsx.



В ходе описания использования было наглядно показан цикл нормирования одной спецификации. Начиная от ввода данных с конструкторской документации и выводом сводной ведомости материалов в формате данных MS Excel.

# ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА ПО РАЗРАБОТКЕ ПРОГРАММНОГО СРЕДСТВА

## Общая характеристика программного средства

Целью дипломного проекта является разработка Системы нормирования материалов по конструкторской документации для любых производств.

Основными пользователями продукта будут являются технологи им система будет помогать считать(нормировать) узлы конструкторских изделий.

Также продуктом будет пользоваться главный технолог. Система нормирования материалов по конструкторской документации позволяит ему контролировать работу технологов, по графику в котором будет отоброжаться информация о посчитаных узлах конструкторского изделия.

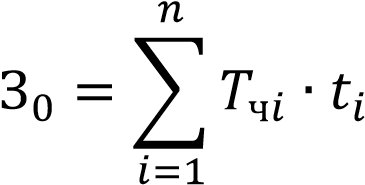
## Расчет затрат на разработку ПО

Выполним упрощенный расчет затрат на разработку ПО в разрезе следующих статей:

* затраты на основную заработную плату разработчиков;
* затраты на дополнительную заработную плату разработчиков;
* отчисления на социальные нужды;
* прочие затраты;
  + 1. **Расчет затрат на основную заработную плату разработчиков**

Затраты на основную заработную плату команды разработчиков определяются исходя из состава и численности команды, размеров месячной заработной платы каждого из участников команды, а также общей трудоемкости разработки программного обеспечения.

Расчет величины заработной платы участников команды осуществляется по формуле:

,

где 𝑛 – количество исполнителей, занятых разработкой конкретного ПО,

𝑇ч𝑖 – часовая заработная плата 𝑖-го исполнителя, 𝑡𝑖 – трудоемкость работ, выполняемых 𝑖-м исполнителем

Часовую заработную плату определим путем деления месячной платы на количество рабочих часов в месяце (168).

Месячную заработную плату определим по фактическим данным предприятия, на котором проходилась преддипломная практика. Для разработки приложения будет задействован программист, месячная заработная плата определена в размере 1500 рублей;

Трудоемкость определим исходя из сложности разработки программного продукта и объема выполняемых им функций. После согласования с руководителем дипломного проекта от выпускающей кафедры трудоемкость програмиста принята в количестве 120 часов;

Также в затраты на основную заработную плату разработчиков входит премиальный фонд. Его размер определим, исходя из практики, сложившейся в организации, в размере 50 % от размера основной заработной платы.

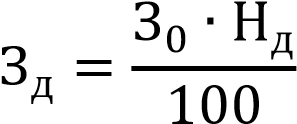
Расчет затрат на основную заработную плату осуществим в форме таблицы.

Таблица 6.1 – Затраты на основную заработную плату

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Участник команды | Выполняемые работы | Месяч ная  зарабо тная  плата, руб. | Часовая заработн ая плата, руб. | Трудоемко сть работ, часов | Основная заработна я плата, руб. |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | Программист | Разработка Развертывание | 1500 | 8,93 | 120 | 1 071,43 |
|  | ПРЕМИЯ | | | | | 50% |
|  | Итого затраты на основную заработную плату разработчиков | | | | | 1 607,14 |

* + 1. **Расчет затрат на дополнительную заработную плату**

Затраты на дополнительную заработную плату включает выплаты, предусмотренные законодательством о труде и определяется по формуле:

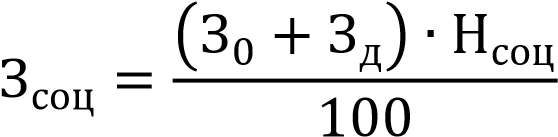
,

где З0 – затраты на основную заработную плату с учетом премии Нд – норматив дополнительной заработной платы, примем в размере 15%

Подставив исходные значения в формулу, получим:

* + 1. **Расчет затрат на социальные нужды**

Отчисления на социальные нужды определяются в соответствии с действующими законодательными актами по формуле:

,

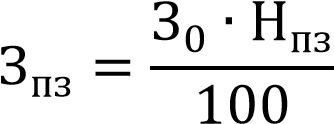
где Нсоц – норматив отчислений на социальные нужды. Согласно действующему законодательству, отчисления в фонд социальной защиты населения и обязательного страхования составляют 34,6 % от фонда основной и дополнительной заработной платы исполнителей.

Подставив исходные значения в формулу, получим:

* + 1. **Расчет прочих затрат**

Дополнительные виды затрат, такие, как оплата консультационных услуг, необходимых для разработки ПО, оплата работ сторонней организации, приобретение права на использование уже существующего ПО и другие, для данного проекта не актуальны.

Также включим в расчет прочие затраты, связанные с разработкой ПО, такие как оплата аренды помещений, освещение, отопление, оплата серверов разработки и т.д. Расчет прочих затрат осуществляется в процентах от затрат на основную заработную плату разработчиков, с учетом премии, по формуле:

,

где Нпз – норматив прочих затрат, примем в размере 120%.

Подставив исходные значения в формулу, получим:

Полную сумму затрат на разработку программного обеспечения находим путем суммирования всех рассчитанных статей затрат. Соберем все данные в таблице.

Таблица 6.2 – Таблица затрат на разработку ПО

|  |  |
| --- | --- |
| Статья затрат | Сумма, руб. |
| Основная заработная плата команды разработчиков | 1607,14 |
| Дополнительная заработная плата команды разработчиков | 241,07 |
| Отчисления на социальные нужды | 639,48 |
| Прочие затраты | 1928,57 |
| Общая сумма затрат на разработку | 4416,27 |

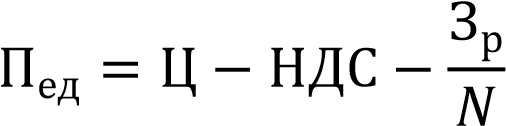
## Оценка результата от продажи ПО. Расчет экономического эффекта.

ПО, разрабатываемое в рамках дипломного проекта, является программным обеспечением для свободной реализации на рынке IT. Поэтому, экономический эффект организации-разработчика в данном случае заключается в получении прибыли от его продажи множеству потребителей. Прибыль от реализации в данном случае напрямую зависит от объемов продаж, цены реализации и затрат на разработку данного ПО.

Таким образом, необходимо сделать обоснование предполагаемого объема продаж – количество подписок на ПС, которое будет куплено клиентами за год. Для расчета необходимы данные некоторого, уже внедренного аналога. В большинстве случаев подобные данные закрыты, однако, ориентируясь на данные о среднем количестве подписок на подобные ПС, можно сделать вывод, что планируемое количество подписок в течение года – около 60.

Определим цену на одну подписку на ПС.

Рассчитаем цену, основываясь на подходе, при котором цена формируется под воздействием спроса и предложения. Расчет прибыли от продажи одной подписки на ПС осуществляется по формуле:



где Ц – цена реализации одной подписки

Зр – сумма расходов на разработку и реализацию

𝑁 – количество копий, купленных за год

НДС – сумма налога на добавленную стоимость

Сумму затрат на реализацию примем в размере 10% от затрат на разработку, соответственно:

Рассчитаем сумму затрат на реализацию и разработку:

Зр = Зразр + Зреал = 4416,27 + 441,63 = 4857,89 руб.

Цену одной подписки обоснуем средней ценой на аналогичное программное обеспечение на рынке. На данны момент цена более сложного аналога ПС на бессрочную лецензию состовляет 750 евро, с учетом того что мы выбрали временную лицензию (подписку) цена подписки будет составляет около 199 рублей.

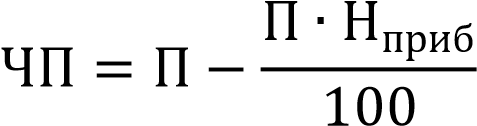
Сумму налога на добавленную стоимость можно рассчитать по формуле:

Подставив вычисленные значения в исходную формулу, получим следующий результат:

Суммарную годовую прибыль по проекту в целом рассчитаем следующим образом:

П = Пед ∙ 𝑁 = 160,98 ∙ 60 = 9658,53 руб.

Чистая прибыль с учетом налога на прибыль рассчитывается по формуле:



Подставив текущее значение налога на прибыль – 18%, получим следующий результат:

## Расчет показателей эффективности инвестиций в разработку ПО

Рассчитаем простую норму (рентабельность инвестиций в разработку и использование ПО):

## Выводы по технико-экономическому обоснованию

В результате технико-экономического обоснования применения программного продукта были получены следующие значения показателей их эффективности:

1. Чистая прибыль от реализации ЧП = 7919,99 руб. остается организации-разработчику и представляет собой экономический эффект от создания нового программного средства.
2. Продукт является экономически выгодным, так как он окупается чуть менее, чем за один год, а рентабельность составляет 163,03%. Таким образом, данная разработка является экономически целесообразной.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта было разработано клиент-серверное приложение, представляющее собой систему нормирования материалов, призванное облегчить, систематизировать и автоматизировать нормирование материалов изделий.

В ходе работы были изучены многие интересные технологии, при-званные облегчить и разнообразить разработку клиент-серверного при-ложения на платформе java. Были подробно изучены Фреймворк JavaFX и база данных MySQL. Так же был изучен и применён шаблон MVC.

На основе анализа, проектирования и расчетов создана Система нормирования материалов для заказчика, которая затем была внедрена в его предприятие. Мое решение позволило предприятию эффективно работать с поступающими заказами.

Разработанная Система нормирования материалов в конструкторской документации уже активно пользуется отдел технологического инжиниринга, что, в дополнение к хорошим отзывам от заказчика, является отличным доказательством востребованности разработки, а также удачности выбранных мной решений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Java 8. Полное руководство, Герберт Шилдт, 2015
2. MySQL — Википедия [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/MySQL>. – Дата доступа: 01.04.2019.
3. JavaFX | Введение [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/javafx/1.1.php>. – Дата доступа: 01.04.2019.
4. NetBeans — Википедия [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/NetBeans>. – Дата доступа: 01.04.2019.
5. Нормирование материалов [Электронный ресурс]. – Режим доступа: https://ascon.ru/products/895/review. – Дата доступа: 01.04.2019.
6. Intermech АРМ материального нормирования [Электронный ресурс]. – Режим доступа: www.intermech.ru/arm\_matnorm.htm. – Дата доступа: 01.04.2019.

ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММНОГО КОДА

Исходный код некоторых контроллеров:

*ParametersController.java*

*package controllers;*

*import com.jfoenix.controls.JFXTextField;*

*import java.net.URL;*

*import java.util.ResourceBundle;*

*import javafx.fxml.FXML;*

*import javafx.fxml.Initializable;*

*import javafx.scene.input.KeyCode;*

*import controllers.MainController.\*;*

*import static controllers.Variables.\*;*

*import models.SpecificationCommonMaterials;*

*import java.sql.SQLException;*

*import java.util.ArrayList;*

*import javafx.application.Platform;*

*import javafx.collections.FXCollections;*

*import javafx.collections.ObservableList;*

*import javafx.scene.Node;*

*import javafx.stage.Stage;*

*public class ParametersController implements Initializable*

*{*

*@FXML*

*private JFXTextField jfxtextfieldLength;*

*@FXML*

*private JFXTextField jfxtextfieldWidth;*

*@FXML*

*private JFXTextField jfxtextfieldNomberOfLayers;*

*double a;*

*double b;*

*double c;*

*double result1;*

*double result2;*

*/\*\**

*\* Initializes the controller class.*

*\*/*

*@Override*

*public void initialize(URL url, ResourceBundle rb)*

*{*

*jfxtextfieldLength.setOnKeyPressed((event) ->*

*{*

*if (event.getCode() == KeyCode.ENTER)*

*{*

*jfxtextfieldWidth.requestFocus();*

*}*

*});*

*jfxtextfieldWidth.setOnKeyPressed((event) ->*

*{*

*if (event.getCode() == KeyCode.ENTER)*

*{*

*jfxtextfieldNomberOfLayers.requestFocus();*

*}*

*});*

*jfxtextfieldNomberOfLayers.setOnKeyPressed((event) ->*

*{*

*if (event.getCode() == KeyCode.ENTER)*

*{*

*try*

*{*

*a = Double.parseDouble(jfxtextfieldLength.getText());*

*b = Double.parseDouble(jfxtextfieldWidth.getText());*

*c = Double.parseDouble(jfxtextfieldNomberOfLayers.getText());*

*result1 = (a \* b \* 0.5 \* c) / 1000000;*

*result2 = (a \* b \* 0.1) / 1000000;*

*if (type == "Bar")*

*{*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа,вспб,вспв, НРМ)\n"*

*+ "VALUES(?,414,4," + a + "," + b + "," + c + "," + result1 + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа, НРМ)\n"*

*+ "VALUES(?,333,4," + result1 + "," + result1 / 5 + ");", idElement);*

*}*

*else*

*{*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа,вспб,вспв, НРМ)\n"*

*+ "VALUES(?,414,4," + a + "," + b + "," + c + "," + result1 + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа, НРМ)\n"*

*+ "VALUES(?,333,4," + result1 + "," + result1 / 5 + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа, НРМ)\n"*

*+ "VALUES(?,78,4, " + a + "," + (a / 1000) \* 2 + ");", idElement);*

*if ((a \* b) / 1000000 > 6.25)*

*{*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа, НРМ)\n"*

*+ "VALUES(?,63,4," + 1 + "," + 1 + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа, НРМ)\n"*

*+ "VALUES(?,415,4," + 0.5 + "," + 0.5 + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа, НРМ)\n"*

*+ "VALUES(?,358,4," + 2 + "," + (0.3 \* 2) + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа,вспб,вспв, НРМ)\n"*

*+ "VALUES(?,1511421945,4," + a + "," + b + "," + 1 + "," + result2 + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа,вспб,вспв, НРМ)\n"*

*+ "VALUES(?,1511421945,8," + a + "," + b + "," + 1 + "," + result2 + ");", idElement);*

*}*

*else*

*{*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа, НРМ)\n"*

*+ "VALUES(?,63,4," + 0.5 + "," + 0.5 + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа, НРМ)\n"*

*+ "VALUES(?,415,4," + 0.25 + "," + 0.25 + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа, НРМ)\n"*

*+ "VALUES(?,358,4," + 1 + "," + (0.3 \* 1) + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа,вспб,вспв, НРМ)\n"*

*+ "VALUES(?,1511421945,4," + a + "," + b + "," + 1 + "," + result2 + ");", idElement);*

*insertCommonElementToDb("INSERT INTO ЭлементыВспомогательные ( IDЭлемента,IDВспомогательного,IDЦехаВ, вспа,вспб,вспв, НРМ)\n"*

*+ "VALUES(?,1511421945,8," + a + "," + b + "," + 1 + "," + result2 + ");", idElement);*

*}*

*}*

*}*

*catch (java.lang.NumberFormatException exception)*

*{*

*Stage stage = (Stage) ((Node) (event.getSource())).getScene().getWindow();*

*stage.close();*

*}*

*Stage stage = (Stage) ((Node) (event.getSource())).getScene().getWindow();*

*stage.close();*

*}*

*});*

*}*

*private void insertCommonElementToDb(String sql, int id)*

*{*

*try*

*{*

*if (connection == null)*

*{*

*System.out.println("Нет соединения с БД!");*

*System.exit(0);*

*}*

*preparedStatement = connection.prepareStatement(sql);*

*preparedStatement.setInt(1, id);*

*preparedStatement.executeUpdate();*

*}*

*catch (SQLException ex)*

*{*

*ex.printStackTrace();*

*}*

*finally*

*{*

*if (preparedStatement != null)*

*{*

*try*

*{*

*preparedStatement.close();*

*}*

*catch (SQLException ex)*

*{*

*}*

*}*

*}*

*}*

*}*

Пример файла представления

SplashScreenView.fxml

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.image.\*?>

<?import java.lang.\*?>

<?import java.util.\*?>

<?import javafx.scene.\*?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<StackPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controllers.SplashScreenController">

<children>

<ImageView pickOnBounds="true" preserveRatio="true" StackPane.alignment="CENTER">

<image>

<Image url="@../resources/background.png" />

</image>

</ImageView>

</children>

</StackPane>

Исходный код некоторых моделей:

Elements.java

package models;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.beans.property.SimpleStringProperty;

/\*\*

\*

\* @author Unlimited

\*/

public class Elements

{

private final SimpleIntegerProperty Элементы\_IDЭлементов;

private final SimpleStringProperty Элементы\_КТДЭ;

private final SimpleStringProperty Элементы\_Обозначение;

private final SimpleStringProperty Элементы\_Наименование;

private final SimpleStringProperty Элементы\_примечание;

public Elements(int Элементы\_IDЭлементов, String Элементы\_КТДЭ, String Элементы\_Обозначение, String Элементы\_Наименование, String Элементы\_примечание)

{

this.Элементы\_IDЭлементов = new SimpleIntegerProperty(Элементы\_IDЭлементов);

this.Элементы\_КТДЭ = new SimpleStringProperty(Элементы\_КТДЭ);

this.Элементы\_Обозначение = new SimpleStringProperty(Элементы\_Обозначение);

this.Элементы\_Наименование = new SimpleStringProperty(Элементы\_Наименование);

this.Элементы\_примечание = new SimpleStringProperty(Элементы\_примечание);

}

public int getЭлементы\_IDЭлементов()

{

return Элементы\_IDЭлементов.get();

}

public void setЭлементы\_IDЭлементов(int value)

{

Элементы\_IDЭлементов.set(value);

}

public SimpleIntegerProperty Элементы\_IDЭлементовProperty()

{

return Элементы\_IDЭлементов;

}

public String getЭлементы\_КТДЭ()

{

return Элементы\_КТДЭ.get();

}

public void setЭлементы\_КТДЭ(String value)

{

Элементы\_КТДЭ.set(value);

}

public SimpleStringProperty Элементы\_КТДЭProperty()

{

return Элементы\_КТДЭ;

}

public String getЭлементы\_Обозначение()

{

return Элементы\_Обозначение.get();

}

public void setЭлементы\_Обозначение(String value)

{

Элементы\_Обозначение.set(value);

}

public SimpleStringProperty Элементы\_ОбозначениеProperty()

{

return Элементы\_Обозначение;

}

public String getЭлементы\_Наименование()

{

return Элементы\_Наименование.get();

}

public void setЭлементы\_Наименование(String value)

{

Элементы\_Наименование.set(value);

}

public SimpleStringProperty Элементы\_НаименованиеProperty()

{

return Элементы\_Наименование;

}

public String getЭлементы\_примечание()

{

return Элементы\_примечание.get();

}

public void setЭлементы\_примечание(String value)

{

Элементы\_примечание.set(value);

}

public SimpleStringProperty Элементы\_примечаниеProperty()

{

return Элементы\_примечание;

}

}

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | | | | *Дополнительные сведения* | |
|  | | | | | **Текстовые документы** | | | |  | |
|  | | | | |  | | | |  | |
| БГУИР ДП 1–40 04 01 00 103 ПЗ | | | | | **Пояснительная записка** | | | | 42 с. | |
|  | | | | | **Отзыв руководителя** | | | |  | |
|  | | | | | **Рецензия** | | | |  | |
|  | | | | | **Акт внедрения** | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | | **Графические документы** | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.400401.103 ПД1 | | | | | **Система нормирования материалов** | | | | Формат А1 | |
|  | | | | | **по КД. Схема структуры** | | | |  | |
|  | | | | | **программы.** | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.400401.103 ПД2 | | | | | **Система нормирования материалов** | | | | Формат А1 | |
|  | | | | | **по КД. Диаграмма классов.** | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.400401.103 ПД3 | | | | | **Система нормирования материалов** | | | | Формат А1 | |
|  | | | | | **по КД. Схема базы данных.** | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.400401.103 ПД4 | | | | | **Система нормирования материалов** | | | | Формат А1 | |
|  | | | | | **по КД. Схема программы.** | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.400401.103 ПЛ1 | | | | | **Система нормирования материалов** | | | | Формат А1 | |
|  | | | | | **по КД. Диаграмма потоков** | | | |  | |
|  | | | | | **данных системы. Плакат.** | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.400401.103 ПЛ2 | | | | | **Система нормирования материалов** | | | | Формат А1 | |
|  | | | | | **по КД. Визуальный интерфейс** | | | |  | |
|  | | | | | **Программного средства.** | | | |  | |
|  | | | | | **Плакат.** | | | |  | |
|  | | | | |  | | | |  | |
|  |  |  |  |  | ***БГУИР ДП 1-40 04 01 00 103 Д1*** | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм.* | *Л.* | *№ докум.* | *Подп.* | *Дата* | ***Система нормирования материалов в конструкторсокй документации.***  ***Ведомость дипломного  проекта.*** | *Лит* | | | *Лист* | *Листов* |
| ***Разраб.*** | | ***Авхимович А.В.*** |  |  |  | *Т* |  | *42* | *42* |
| ***Пров.*** | | ***Жвакина А.В.*** |  |  | ***ИиТП***  ***гр. 493551*** | | | | |
| ***Т.контр.*** | | ***Летохо А.С.*** |  |  |
| ***Н.контр.*** | | ***Шиманский В.В.*** |  |  |
| ***Утв.*** | | ***Волорова Н.А.*** |  |  |