Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных технологий

Кафедра проектирования информационных компьютерных систем

Дисциплина "Современные технологии проектирования информационных систем"

К защите допустить:
Руководитель курсовой работы
старший преподаватель
кафедры
_____ А.В.Михалькевич
11.07.2025

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Документация структурного производственного подразделения

БГУИР КР 1-40 05 01-10 № 184 ПЗ

Студент (подпись студента)

Курсовая работа
представлена на проверку
11.07.2025
_____
(подпись студента)

2025

Реферат

БГУИР КР 1-40 05 01-10 № 184 ПЗ, гр. 784371

, Документация структурного производственного подразделения, Минск: БГУИР - 2025.

Пояснительная записка 79394 с., 14 рис., 0 табл.

Ключевые слова:

Предмет Современные технологии проектирования информационных систем, А.В.Михалькевич

-

-

Содержание

Введение

Веб-приложения – это специальный вид приложений, которые работают в глобальной сети Интернет или внутренней локальной сети по протоколу HTTP(S). Как правило, веб-приложения не требуют установки дополнительного программного обеспечения на стороне клиента, а вся логика, в основном, выполняется на стороне сервера. Для отображения пользовательского интерфейса используется браузер – программа, способная распознавать язык разметки HTML (и сопутствующие технологии – таблицы стилей CSS, клиентский скриптовой язык программирования JavaScript и т.д.). Для взаимодействия клиента и сервера используется протокол HTTP, который работает по схеме «запрос-ответ». В момент, когда клиент хочет обратиться к серверу, он генерирует запрос, который отправляется серверу. Сервер обрабатывает этот запрос и подготавливает ресурсы, которые будут отправлены клиенту. После этого сервер генерирует ответ, в котором содержаться все необходимые данные и отправляет клиенту. Работа веб-приложений заключается в формировании необходимых данных как раз в момент подготовки ресурсов на сервере. Обычно в этот момент запускается некоторый программный код, который содержит определенную бизнес логику. По сравнению с настольными приложениями, веб-приложения обладают более ограниченными возможностями по формированию пользовательского интерфейса и клиентской функциональности. Развитие веб-технологий доказало, что веб-приложения также могут реализовывать богатые сценарии и

успешно конкурировать с настольными приложениями. Кроме того, за последние несколько лет очень активно развиваются технологии, позволяющие сделать веб-приложения еще более интерактивными. Основная цель данного курсового проекта – создание веб-приложения с организацией взаимодействия с базой данных на объектно-ориентированном языке Java. Для достижения цели была поставлена задача создать систему учета документооборота производственного подразделения. Уровень сложности этой задачи позволяет в достаточной мере ознакомиться с основными этапами разработки веб-приложений и приобрести определенные навыки программирования.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Производственное подразделение – самостоятельное структурное подразделение, выполняющее определенные функции в системе управления предприятием. Обычно таким подразделением является финансовый отдел, планово-производственный отдел и др. Его структура и численность зависит от организационно-правовой формы предприятия, характера финансовой деятельности, объема производства, количества работающих на предприятии.

Производственное подразделение является частью единого механизма управления хозяйственной деятельностью, и поэтому оно тесно связано с другими службами предприятия. Так, в результате тесных контактов с бухгалтерией, структурному подразделению предприятиия представляются планы производства, списки кредиторов и дебиторов, документы по выплате зарплаты работникам, суммах денежных средств, находящихся на его счетах, и суммах предстоящих расходов. В свою очередь планово-финансовый отдел, обрабатывая эту информацию, анализируя ее, дает квалифицированную оценку платежеспособности предприятия, ликвидности его активов, кредитоспособности, составляет платежный календарь, готовит аналитические отчеты по другим параметрам финансового состояния предприятия и знакомит бухгалтерию с финансовыми планами и аналитическими отчетами об их выполнении, которая в своей ежедневной деятельности руководствуется этими сведениями.

От отдела маркетинга производственное подразделение получает планы по сбыту продукции и использует ее при планировании доходов и составлении оперативных финансовых планов. Для проведения успешной маркетинговой компании производственное подразделение обосновывает реализационные цены, утверждает систему уступок в цене контракта, анализирует сбытовые и маркетинговые издержки, осуществляет сравнительную оценку конкурентоспособности продукции предприятия, оптимизирует ее рентабельность, создавая, таким образом, условия для заключения крупных сделок.

Основными задачами структурных производственных подразделений в организации являются:

- финансирование затрат на производство продукции, капитальных вложений и других расходов;
- выполнение финансовых обязательств перед госбюджетом, банками, поставщиками, вышестоящими организациями, рабочими и служащими;
- изыскание путей увеличения денежных доходов, прибыли и повышения рентабельности;
- обеспечение сохранности оборотных средств и ускорение их оборачиваемости;
- осуществление контроля за эффективным использованием основных производственных

фондов и инвестиций;
организация совершенствование форм денежных расчетов с поставщиками, покупателями, рабочими и служащими, вышестоящей организацией, госбюджетом, банками.

Финансовый отдел получает от бухгалтерии, планового отдела, отделов снабжения и сбыта, технического отдела, отдела капитального строительства, отдела главного механика и других структурных подразделений и служб необходимые плановые и отчетные материалы; бухгалтерские отчеты и балансы; проекты и утвержденные планы выпуска товарной продукции и ее себестоимости, реализации продукции и прибыли; планы движения ТМЦ (производственных запасов, незавершенного производства и готовой продукции); расчеты экономической эффективности капитальных вложений и новой техники, технико-экономические показатели; сметы затрат на производство и расходование спецсредств.

Организация управление документооборотом – одна из отраслей управленческой деятельности, поэтому среди руководителей предприятий и их структурных подразделений становятся актуальными и приобретают широкое распространение решения для автоматизации документооборота. В настоящее время появилось немало информационных программных систем для управления документооборотом, значительно упрощающих все операции документооборота. Данные системы обеспечивают автоматизацию документооборота, электронную обработку документов и предоставляют эффективные механизмы организации документооборота на предприятии и управления документооборотом.

2 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

Решение поставленной задачи основывается на анализе процессов регистрации и учета документов, поступающих в планово-производственную службу предприятия, а также выделении из них тех задач, которые можно автоматизировать путем создания веб-приложения с доступом к базе данных.

В ходе реализации серверной части приложения принято решение организовать взаимодействие с базой данных MySQL-сервера посредством драйвера, предоставленного производителем СУБД.

В процессе разработки сервера поставлена задача реализовать функции отправки/получения данных из СУБД, их консолидации и формирования веб-страниц, отображаемых в браузере пользователя.

Библиотека не только решает задачу связи классов Java с таблицами базы данных (и типов данных Java с типами данных SQL), но и также предоставляет средства для автоматической генерации и обновления набора таблиц, построения запросов и обработки полученных данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание SQL- и JDBC-кода.

Hibernate автоматизирует генерацию SQL-запросов и освобождает разработчика от ручной обработки результирующего набора данных и преобразования объектов, максимально облегчая перенос (портирование) приложения на любые базы данных SQL.

Hibernate обеспечивает прозрачную поддержку сохранности данных для POJO-объектов (стандартных Java-объектов, не унаследованных от какого-то специфического объекта и не реализующих никаких служебных интерфейсов сверх тех, которые нужны для бизнес-модели).

Сопоставление Java-классов с таблицами базы данных осуществляется с помощью конфигурационных XML-файлов или Java-аннотаций. При использовании файла XML Hibernate может генерировать скелет исходного кода для классов длительного хранения.

Hibernate поддерживает отображение пользовательских типов значений. Это делает возможными такие сценарии:

переопределение типа по умолчанию SQL, Hibernate выбирает при отображении столбца свойства.
проецирование перечисляемого типа Java на поле БД, будто они являются обычными свойствами.
проецирование одного свойства в несколько колонок.

Hibernate обеспечивает использование SQL-подобного языка Hibernate Query Language (HQL), который позволяет выполнять SQL-подобные запросы, записанные рядом с объектами данных Hibernate.

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ СИСТЕМЫ НА ОСНОВЕ СТАНДАРТА IDEF0

IDEF0 – методология функционального моделирования. С помощью наглядного графического языка IDEF0 изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков – в терминах IDEF0). Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы.

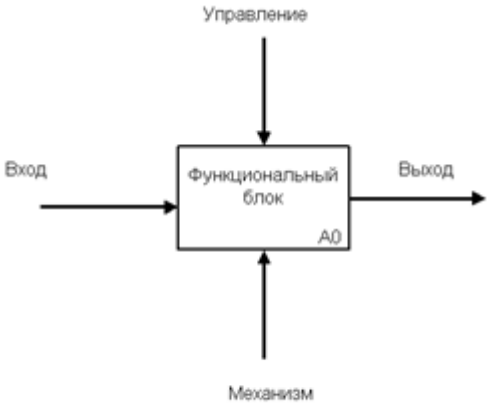
В основе методологии лежат четыре основных понятия:

Первым из них является понятие функционального блока (*Activity Box*). Функциональный блок графически изображается в виде прямоугольника (см. рис. 3.1) и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении.

Каждая из четырех сторон функционального блока имеет своё определенное значение (роль), при этом:

Верхняя сторона имеет значение «Управление» (Control);
Левая сторона имеет значение «Вход» (Input);
Правая сторона имеет значение «Выход» (Output);
Нижняя сторона имеет значение «Механизм» (Mechanism).

Каждый функциональный блок в рамках единой рассматриваемой системы должен иметь свой уникальный идентификационный номер.



Функциональный блок IDEF0

Вторым «китом» методологии IDEF0 является понятие интерфейсной дуги (*Arrow*). Также интерфейсные дуги часто называют потоками или стрелками. Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображенную данным функциональным блоком.

Графическим отображением интерфейсной дуги является однонаправленная стрелка. Каждая интерфейсная дуга должна иметь свое уникальное наименование (*Arrow Label*). По требованию стандарта, наименование должно быть оборотом существительного.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т.д.) или потоки данных и информации (документы, данные, инструкции и т.д.).

В зависимости от того, к какой из сторон подходит данная интерфейсная дуга, она носит название «входящей», «исходящей» или «управляющей». Кроме того, «источником» (началом) и «приемником» (концом) каждой функциональной дуги могут быть только функциональные блоки, при этом «источником» может быть только выходная сторона блока, а «приемником» любая из трех оставшихся.

Третьим основным понятием стандарта IDEF0 является декомпозиция (*Decomposition*). Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели.

Декомпозиция позволяет постепенно и структурированно представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Модель IDEF0 всегда начинается с представления системы как единого целого – одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой, и обозначается идентификатором «A-0».

В процессе декомпозиции, функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие

главные подфункции функционального блока контекстной диаграммы и называется дочерней (*Child diagram*) по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме соответственно называется дочерним блоком – *Child Box*). В свою очередь, функциональный блок – предок называется родительским блоком по отношению к дочерней диаграмме (*Parent Box*), а диаграмма, к которой он принадлежит – родительской диаграммой (*Parent Diagram*). Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока. Важно отметить, что в каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок, или исходящие из него фиксируются на дочерней диаграмме. Этим достигается структурная целостность IDEF0 – модели.

Последним из понятий IDEF0 является глоссарий (*Glossary*). Для каждого из элементов IDEF0: диаграмм, функциональных блоков, интерфейсных дуг существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект, отображенный данным элементом. Этот набор называется глоссарием и является описанием сущности данного элемента.

IDEF0 модели разрабатываемой системы изображены на рис.А.1-А.5 приложения А.

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ

Информационная модель – модель объекта, представленная в виде информации, описывающей существенные для данного рассмотрения параметры и переменные величины объекта, связи между ними, входы и выходы объекта и позволяющая путём подачи на модель информации об изменениях входных величин моделировать возможные состояния объекта.

Информационная модель (в широком, общенаучном смысле) – совокупность информации, характеризующая существенные свойства и состояния объекта, процесса, явления, а также взаимосвязь с внешним миром.

IDEF1X – это методология семантического моделирования данных. Она разработана с учетом следующих требований:

1. *Поддерживает разработку концептуальных схем*

Синтаксис IDEF1X поддерживает семантические конструкции, необходимые для разработки концептуальной схемы. Окончательная версия IDEF1X-модели обладает желаемыми характеристиками – непротиворечивостью, расширяемостью и адаптируемостью.

1. *Обеспечивает ясный язык*

IDEF1X имеет простую, ясную, непротиворечивую структуру и четкие семантические понятия. Синтаксис и семантика IDEF1X сравнительно легки для понимания, хотя и являются достаточно мощным средством.

1. *Проста для изучения*

Семантическое моделирование данных - новое понятие для многих пользователей IDEF1X. Проблема обучаемости этому языку является важным факторомом. Язык рассчитан на понимание и использование как профессиональными бизнесменами и системными аналитиками, так и администраторами данных и разработчиками баз данных. Он может служить эффективным средством коммуникации в коллективах, состоящих из различных специалистов.

1. *Надежно проверена на практике*

IDEF1X базируется на многолетнем опыте предшествующих методологий и тщательно проверена как в проектах ВВС, так и в промышленности.

1. *Возможность автоматизации*

IDEFlX-диаграммы могут создаваться большим числом графических программных пакетов. ВВС США на основе концептуальной схемы разработали активный трехсхемный словарь для построения прикладных программ и обработки запросов в распределенной неоднородной среде. Существует также коммерческое программное обеспечение, поддерживающее детализацию, анализ и управление конфигурацией IDEFlX-моделей.

Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято.

Для разработки оптимальной структуры реляционной БД необходимо проанализировать составляющие ее элементы и отношения между ними, а затем создать нормализованные таблицы. Нормализация таблиц базы данных – первый шаг на пути проектирования структуры реляционной базы данных. База данных считается нормализованной, если ее таблицы представлены как минимум в третьей нормальной форме.

Главная цель нормализации базы данных – устранение избыточности и дублирования информации. При нормализации необходимо, чтобы любое значение хранилось в базе в одном экземпляре. Как следствие, значительно сокращается вероятность появления противоречивых данных, облегчается администрирование базы и обновление информации в ней, сокращается объем дискового пространства.

В ходе разработки структуры базы данных проектируемой системы все таблицы были приведены к 3 нормальной форме, т.е. все они отвечают следующим требованиям:

запрет на повторяющиеся столбцы (содержащие одинаковую по смыслу информацию);
запрет на множественные столбцы (содержащие значения типа список и т.п.);
обязательное определение первичного ключа для таблицы;
зависимость неключевых столбцов таблиц от первичного ключа в целом, но не от его части;
независимость неключевых столбцов от других неключевых столбцов, а зависимость только от первичного ключа.

а) Сущность document– содержит в себе информацию о документе. Таблица важна для того, чтобы узнать характеристики документа: кто инициатор, название документа, описание, дата регистрации, исполнитель, дата готовности, статус документа, входящий номер.

б) Сущность performer – содержит информацию об исполнителях документов: должность, имя, телефон, ник.

в) Сущность type – тип документа.

г) Сущность initiator – содержит информацию об инициаторе документа (отдел предприятия).

д) Сущность status – это таблица, содержащая информацию о статусе документа.

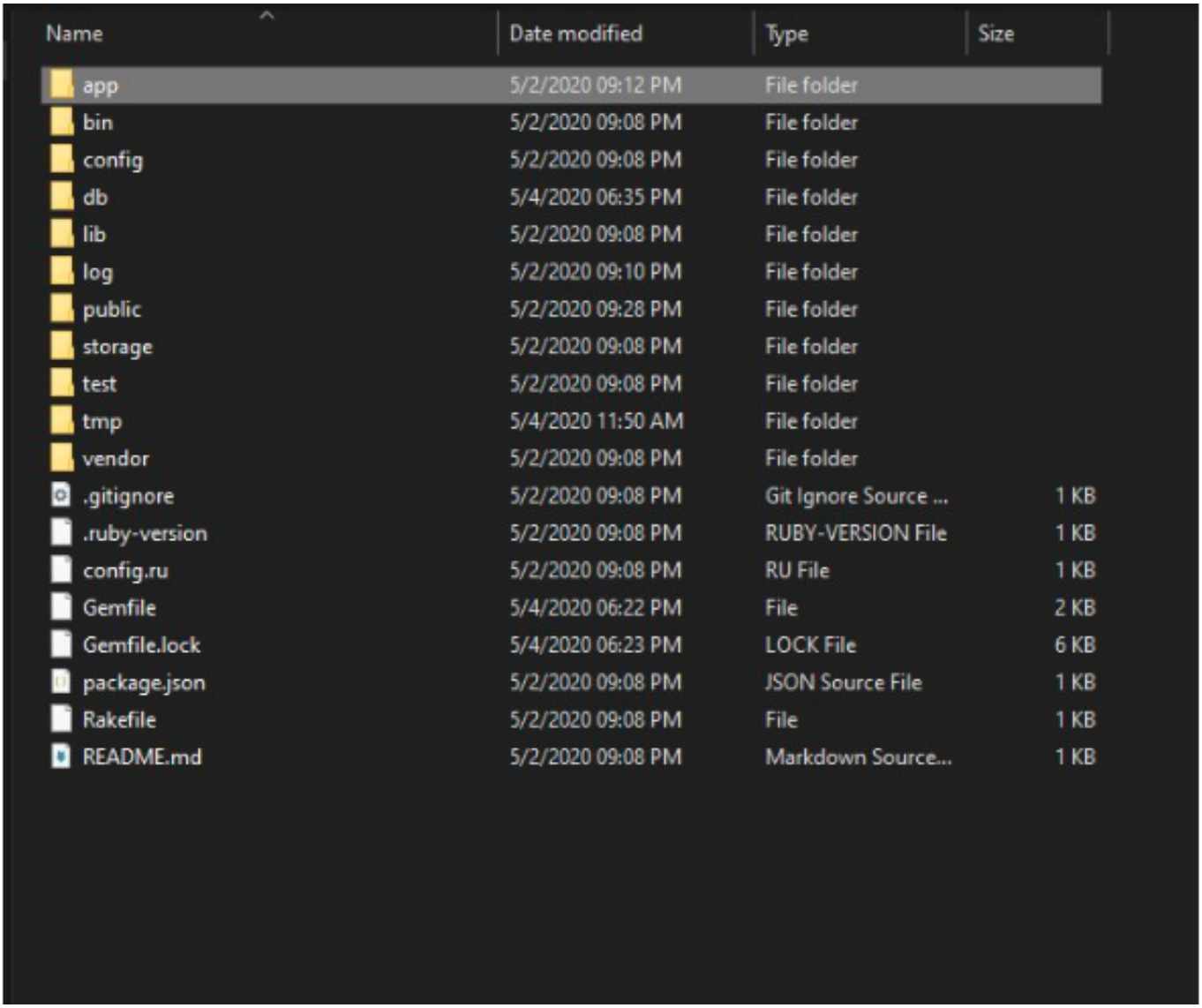


Рисунок 4.1 – Информационная модель проектируемой системы

5 РАЗРАБОТКА МОДЕЛЕЙ ПРОЕКТИРУЕМОЙ СИСТЕМЫ НА ОСНОВЕ СТАНДАРТА UML 2.0

Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования

компонентов программного обеспечения, бизнес-процессов и других систем. Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения.

С помощью UML можно разработать детальный план создаваемой системы, содержащей не только ее концептуальные элементы, такие как системные функции и бизнес-процессы, но и конкретные особенности, например классы, написанные на специальных языках программирования, схемы баз данных и программные компоненты многократного использования.

Язык UML предназначен для описания моделей. На UML можно содержательно описывать классы, объекты и компоненты в различных предметных областях, часто сильно отличающихся друг от друга.

Как любой язык, UML состоит из словаря и правил, позволяющие комбинировать входящие слова и получать осмысленные конструкции.

Моделирование необходимо для понимания системы. Обычно при этом единственной модели никогда не бывает достаточно. Поэтому приходиться разрабатывать большое количество взаимосвязанных моделей.

Словарь в UML включает 3 вида основных конструкций:

Сущности – абстракции, являющиеся основными элементами модели.
Отношения – связи между сущностями.
Диаграммы – группируют множество сущностей и отношений.
В языке UML определены четыре типа отношений:
зависимость;
ассоциация;
обобщение;
реализация.

Эти отношения являются основными связующими строительными блоками в UML и применяются для создания корректных моделей.

Зависимость *(Dependency*) – это семантическое отношение между двумя сущностями, при котором изменение одной из них, независимой, может повлиять на семантику другой, зависимой.

Ассоциация (*Association*) – структурное отношение, описывающее совокупность связей; связь - это соединение между объектами.

Обобщение (*Generalization*) – отношение «специализация/обобщение», при котором объект специализированного элемента (потомок) может быть подставлен вместо объекта обобщенного элемента.

Реализация (*Realization*) – это семантическое отношение между классификаторами, при котором один классификатор определяет «контракт», а другой гарантирует его выполнение

Диаграммы UML – графическое представление набора элементов, изображенное чаще всего в виде связанного графа с вершинам (сущностями) и ребрами (отношениями).

В ходе выполнения курсового проектирования осуществлена разработка следующих моделей системы на основе стандарта UML 2.0.

Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем. Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения.

С помощью UML можно разработать детальный план создаваемой системы, содержащей не только ее концептуальные элементы, такие как системные функции и бизнес-процессы, но и конкретные особенности, например классы, написанные на специальных языках программирования, схемы баз данных и программные компоненты многократного использования.

Язык UML предназначен для описания моделей. На UML можно содержательно описывать классы, объекты и компоненты в различных предметных областях, часто сильно отличающихся друг от друга.

Как любой язык, UML состоит из словаря и правил, позволяющие комбинировать входящие слова и получать осмысленные конструкции.

Моделирование необходимо для понимания системы. Обычно при этом единственной модели никогда не бывает достаточно. Поэтому приходиться разрабатывать большое количество взаимосвязанных моделей.

Словарь в UML включает 3 вида основных конструкций:

Сущности – абстракции, являющиеся основными элементами модели.
Отношения – связи между сущностями.
Диаграммы – группируют множество сущностей и отношений.
В языке UML определены четыре типа отношений:
зависимость;
ассоциация;
обобщение;
реализация.

Эти отношения являются основными связующими строительными блоками в UML и применяются для создания корректных моделей.

Зависимость *(Dependency*) – это семантическое отношение между двумя сущностями, при котором изменение одной из них, независимой, может повлиять на семантику другой, зависимой.

Ассоциация (*Association*) – структурное отношение, описывающее совокупность связей; связь - это соединение между объектами.

Обобщение (*Generalization*) – отношение «специализация/обобщение», при котором объект специализированного элемента (потомок) может быть подставлен вместо объекта обобщенного элемента.

Реализация (*Realization*) – это семантическое отношение между классификаторами, при котором один классификатор определяет «контракт», а другой гарантирует его выполнение

Диаграммы UML – графическое представление набора элементов, изображенное чаще всего в виде связанного графа с вершинам (сущностями) и ребрами (отношениями).

В ходе выполнения курсового проектирования осуществлена разработка следующих моделей системы на основе стандарта UML 2.0.

5.1Диаграмма вариантов использования (прецедентов) (use case diagram)

Суть диаграммы вариантов использования (Приложение Б, рис. Б.1) состоит в следующем: проектируемая система представляется в форме так называемых вариантов использования, с которыми взаимодействуют некоторые внешние сущности или актеры. При этом актером или действующим лицом называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой и собственно выполнение вариантов использования.

5.2 Диаграмма классов (class diagram)

Диаграмма классов (Приложение Б, рис. Б.2-Б.3) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов может служить дальнейшим развитием концептуальной модели проектируемой системы.

5.3 Диаграмма последовательности (sequence diagram)

С помощью диаграммы последовательности (Приложение Б, рис. Б.4) можно описать полный контекст взаимодействий как своеобразный временной график «жизни» всей совокупности объектов, взаимодействующих между собой для реализации варианта использования программной системы, достижения бизнес-цели или выполнения какой-либо задачи. На диаграмме последовательности также изображаются объекты, которые непосредственно участвуют во взаимодействии, при этом никакие статические связи с другими объектами не визуализируются. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени.

5.4 Диаграмма состояний (statechart diagram)

Главное предназначение диаграммы состояний (рисунок 5.1) – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение моделируемой системы в течение всего ее жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий.

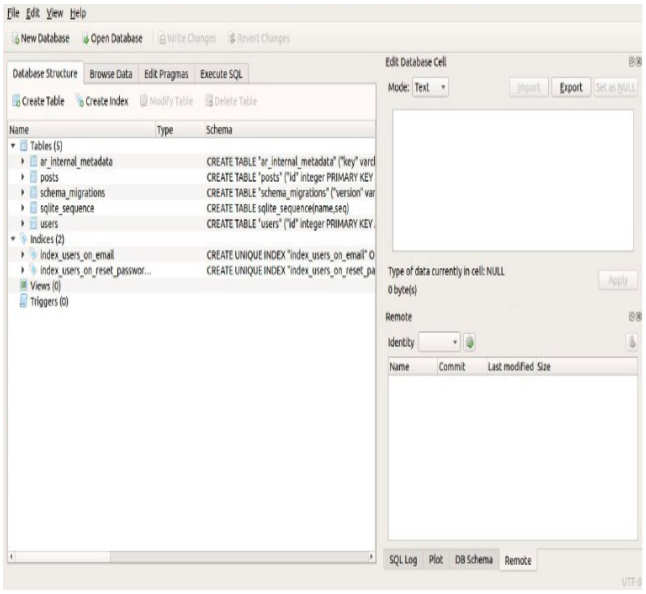


Рисунок 5.1 – Диаграмма состояний

5.5Диаграмма компонентов (component diagram)

Диаграмма компонентов (рисунок 5.2) описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

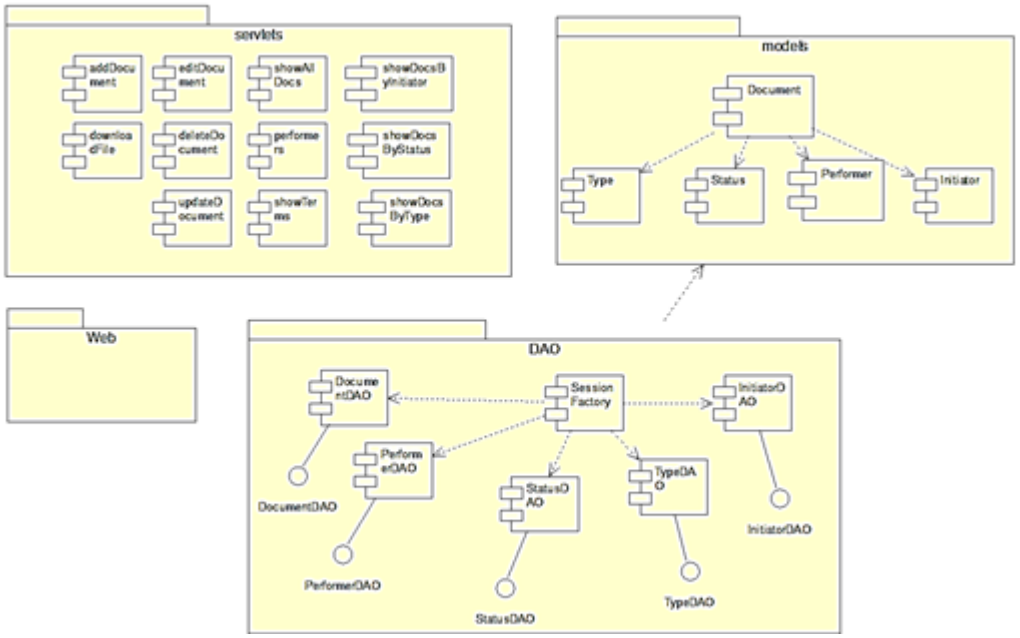


Рисунок 5.2 – Диаграмма компонентов

Диаграмма компонентов разрабатывается для следующих целей:

визуализация общей организации структуры исходного кода программной системы;
спецификация исполнимого варианта программной системы;
обеспечение многократного использования отдельных фрагментов программного кода;
представление концептуальной и физической схем баз данных.

5.6 Диаграмма развертывания (deployment diagram)

Диаграмма развертывания (рисунок 5.3) применяется для представления общей конфигурации и топологии распределенной программной системы и содержит изображение размещения компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений – маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

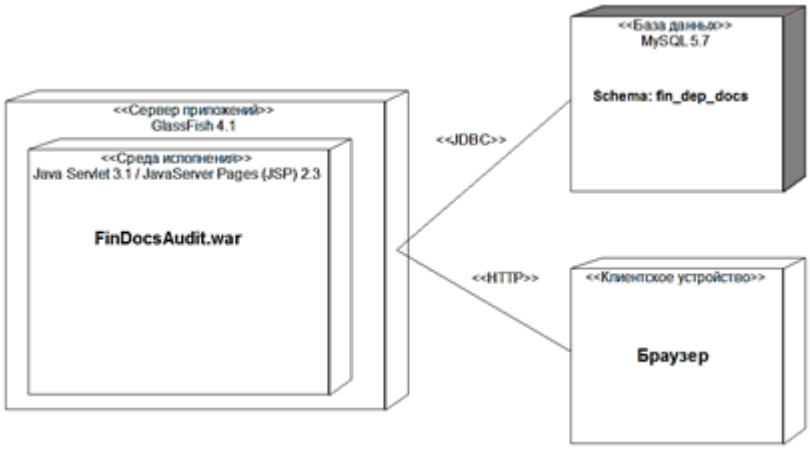


Рисунок 5.3 – Диаграмма развертывания

6 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

На рисунке 6.1 изображена блок-схема алгоритма разрабатываемой функции получения списка документов по идентификатору инициатора. Алгоритмы функций получения списка документов по идентификаторам типов документов, исполнителей и статусов аналогичны приведенному ниже.

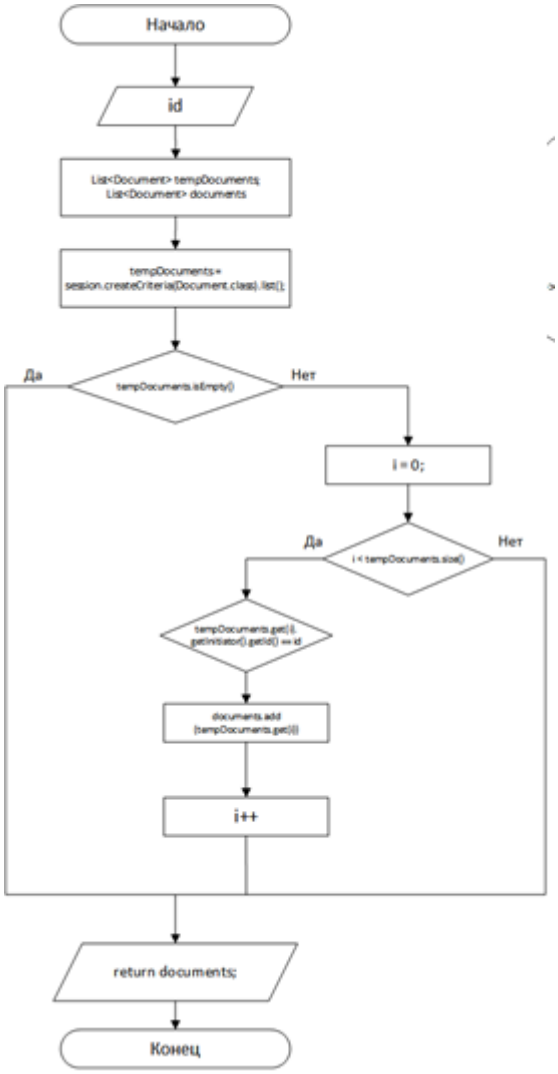


Рисунок 6.1 – Блок-схема функции получения списка документов по идентификатору

Запросы к базе данных осуществляются при помощи инструментария фреймворка Hibernate 4.3, содержащего механизмы отображения в реляционной базе данных объектов Java. Данная библиотека предназначенная для решения задач объектно-реляционного отображения (ORM). Целью Hibernate является освобождение разработчика от значительного объёма сравнительно низкоуровневого программирования при работе в объектно-ориентированных средствах в реляционной базе данных. Разработчик может использовать Hibernate как в процессе проектирования системы классов и таблиц «с нуля», так и для работы с уже существующей базой данных.

7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

7.1 Назначение программы

Приложение «Документация структурного производственного подразделения» (далее – Приложение), разработано для организации работы сотрудников отдела по регистрации, учету и контролю исполнения входящих документов.

Приложение предназначено для следующих целей:

ведение базы данных входящих документов (добавление, редактирование, удаление информации о документах, поступивших на исполнение в планово-производственный

отдел);
загрузка на сервер электронных версий документов;
загрузка с сервера ранее загруженных файлов документов;
отображение статистики по документам, находящихся на исполнении у конкретного сотрудника планово-производственного отдела;
отображение актуальной информации о количестве документов с нарушением контрольных сроков исполнения, находящихся на исполнении у конкретного сотрудника планового-производственного отдела;
отображение актуальной информации о количестве документов с приближающимся контрольным сроком исполнения.

7.2 Условия выполнения

Для работы с клиентской частью Приложения необходимо наличие браузера MS Edge или MS Internet Explorer, Opera, Mozilla Firefox, Google Chrome, Safari с включенной поддержкой JavaScript.

Серверная часть Приложения предназначена для работы на серверах предприятия и должно отвечать следующим минимальным техническим и программным характеристикам:

процессор – Pentium 2 266 МГц и выше;
свободное место на жестком диске – не менее 1 Гб (250 Мб – для JDK8, около 250 Мб – для сервера приложений GlassFish 4.1, около 500 Мб – для СУБД MySQL);
свободная оперативная память – не менее 512 Мб;
ОС MS Windows Vista SP2+ (Windows Server 2008 R2+), Red Hat Enterprise Linux 5.5+, Oracle Linux 5.5+, Suse Linux Enterprise Server 10 SP2+, Ubuntu Linux 12.04 LTS+, Mac OS X 10.8.3+;
программная платформа JDK версии не ниже 8.0;
СУБД MySQL 5.5+;

7.3 Работа с веб-приложением

Запуск веб-приложения осуществляется путем ввода пользователем в адресную строку браузера адреса, порта и корневого контекстного адреса приложения (context-root).

Примечание.

В случае применения настроек по умолчанию и инсталляции сервера приложений на локальный хост веб-адресом приложения будет являться: http://localhost:8080/web/.

В случае корректной настройки веб-приложения и СУБД в браузере отобразится веб-страница, представленная на рисунке 7.1:

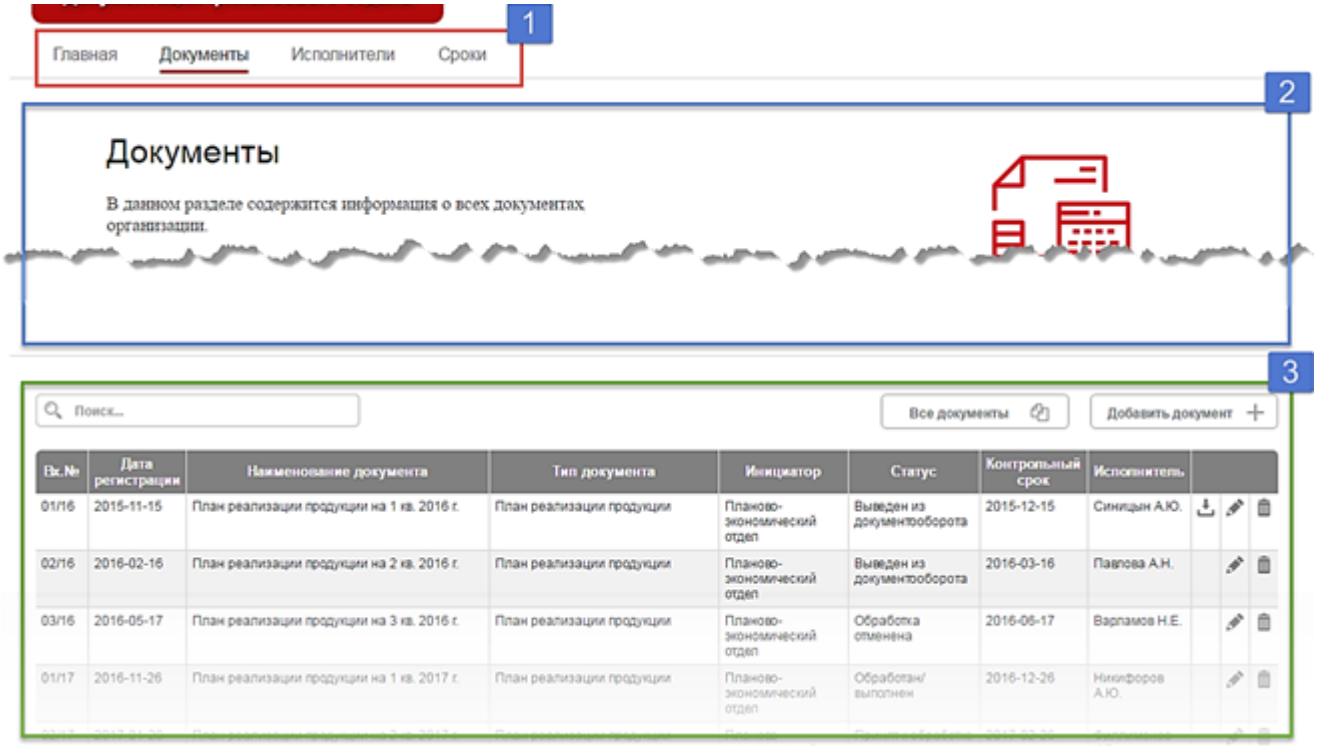


Рисунок 7.1 – Общий вид информационной веб-страницы

Все информационные веб-страницы приложения стилизованы по общим параметрам и состоят из трех областей: области навигационного меню (п.1, выделена красным цветом, содержит гиперссылки на другие веб-страницы приложения), области описания страницы (п.2, выделена синим цветом, содержит заголовок страницы и краткое описание предполагаемых действий пользователя), области отображения рабочего контента (п.3, выделена зеленым цветом, содержит веб-элементы для интерактивного взаимодействия с пользователем).

Все веб-страницы работы с формами (добавление, удаление, редактирование информации) стилизованы по общим параметрам и состоят из четырех областей (см. рисунок 7.2): (п.1, область выделена красным цветом, содержит гиперссылки на другие веб-страницы приложения), области описания страницы (п.2, выделена синим цветом, содержит заголовок страницы и краткое описание предполагаемых действий пользователя), области форм (п.3, выделена зеленым цветом, содержит веб-формы для внесения данных пользователем), дополнительной области навигации (п.4, область выделена оранжевым цветом, содержит гиперссылки на другие веб-страницы приложения или кнопки отправки данным форм).

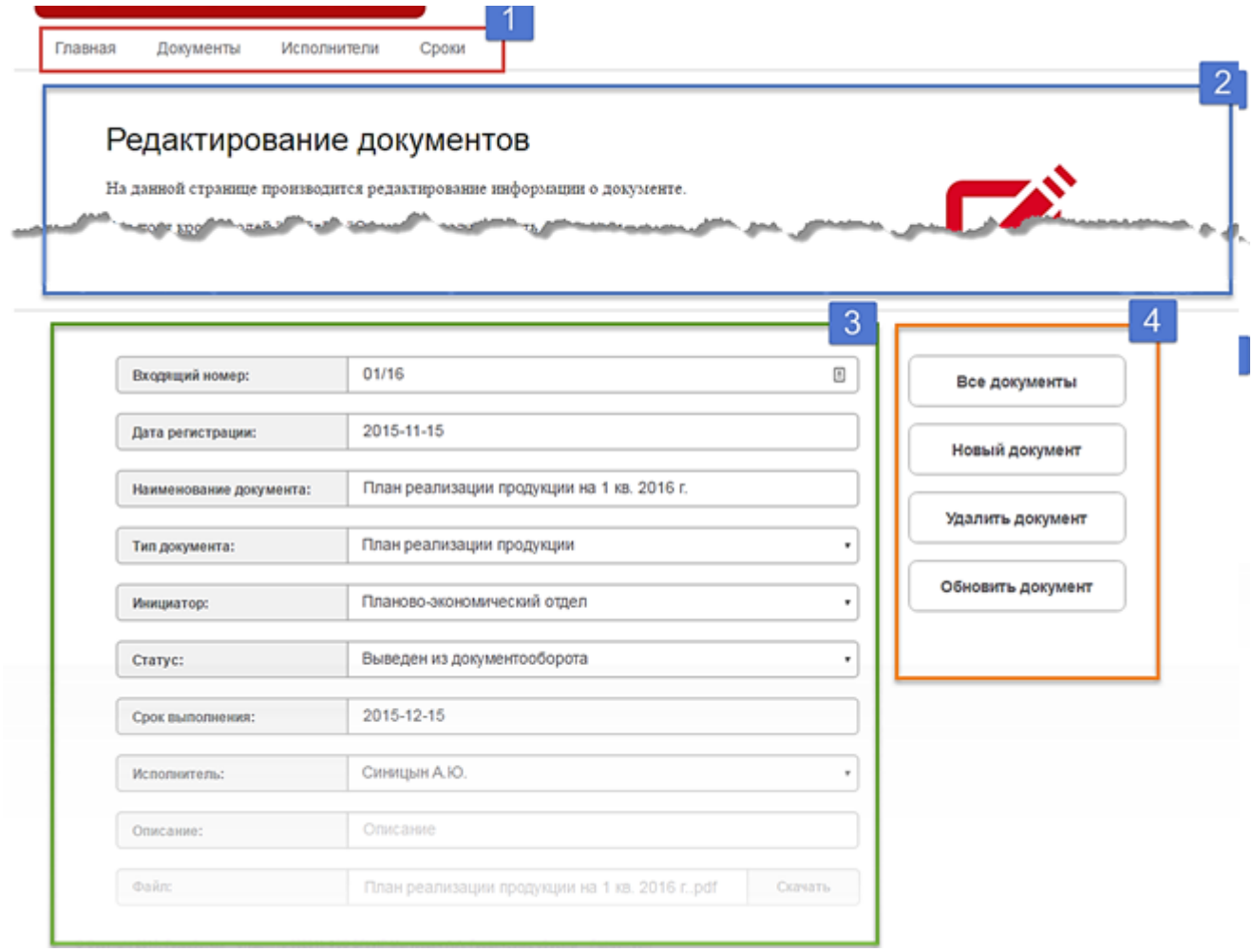


Рисунок 7.2 – Общий вид веб-страницы работы с файлами

Начальной страницей работы веб-приложения является веб-страница «Документы» (см. рисунок 7.3). Далее по тексту будут приводиться снимки только интересующих нас областей.

На данной странице отображена информация о всех зарегистрированных документах, независимо от присвоенного им статуса и контрольного срока. Изначально строки таблицы отсортированы по идентификаторам документов, хранящихся в базе данных, однако предусмотрена сортировка по выбранному критерию – для этого пользователю необходимо нажать на любой из заголовков таблицы (п.1, рисунок 7.3, область выделена синим цветом).

С целью быстрого доступа к странице редактирования конкретного документа пользователю необходимо нажать на интересующую его ячейку столбца «Наименование документа» (п.2, рисунок 7.3, область выделена красным цветом).

С целью осуществления выборки документов по конкретному критерию пользователю необходимо нажать на соответствующее поле таблицы с интересующим его значением. Для выборки доступны любое из полей столбцов «Тип документа», «Инициатор» и «Статус» (п.3, рисунок 7.3, область выделена зеленым цветом).

С целью быстрого доступа к странице конкретного исполнителя пользователю необходимо нажать на интересующую его ячейку столбца «Исполнитель» (п.4, рисунок 7.3, область выделена оранжевым цветом).

Для быстрого доступа к определенным действиям с документами пользователю необходимо нажать на соответствующую пиктограмму (п.5, рисунок 7.3, выделена желтым цветом).

Доступны следующие быстрые действия (перечислены в порядке их отображения): «Загрузить с сервера электронный файл документа», «Редактировать документ» и «Удалить документ». Отсутствие пиктограммы «Загрузить с сервера электронный файл документа» означает отсутствие на сервере загруженного файла документа.

Для быстрого доступа к странице добавления документа необходимо нажать на кнопку «Добавить документ» (п.6, рисунок 7.3).

Для поиска содержимого таблиц по любому из критериев пользователю необходимо воспользоваться соответствующей поисковой строкой (п.7, рисунок 7.3).

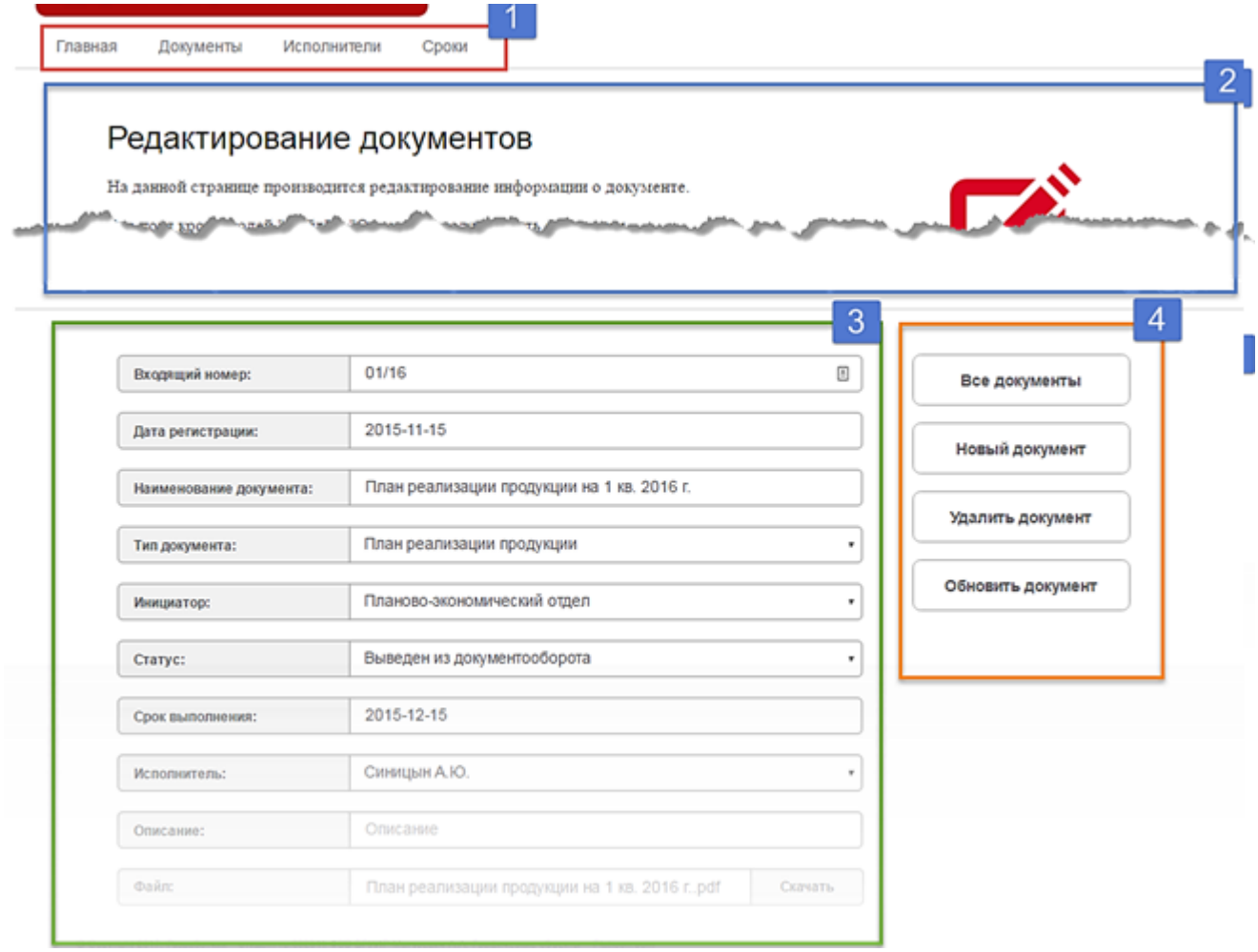


Рисунок 7.3 – Общий вид веб-страницы «Документы»

На странице «Исполнители» (см. рисунок 7.4) отображена информация о конкретном исполнителе и документах, находящихся у него на исполнении. Данная страница содержит четыре области: область информации об исполнителе (п.1, рисунок 7.4, выделена красным цветом), область информации о документах, находящихся на исполнении у конкретного исполнителя (п.2, рисунок 7.4, выделена синим цветом), область статистики по исполнителям (п.3, рисунок 7.4, выделена зеленым цветом), область поиска конкретного исполнителя (п.4, выделена оранжевым цветом).

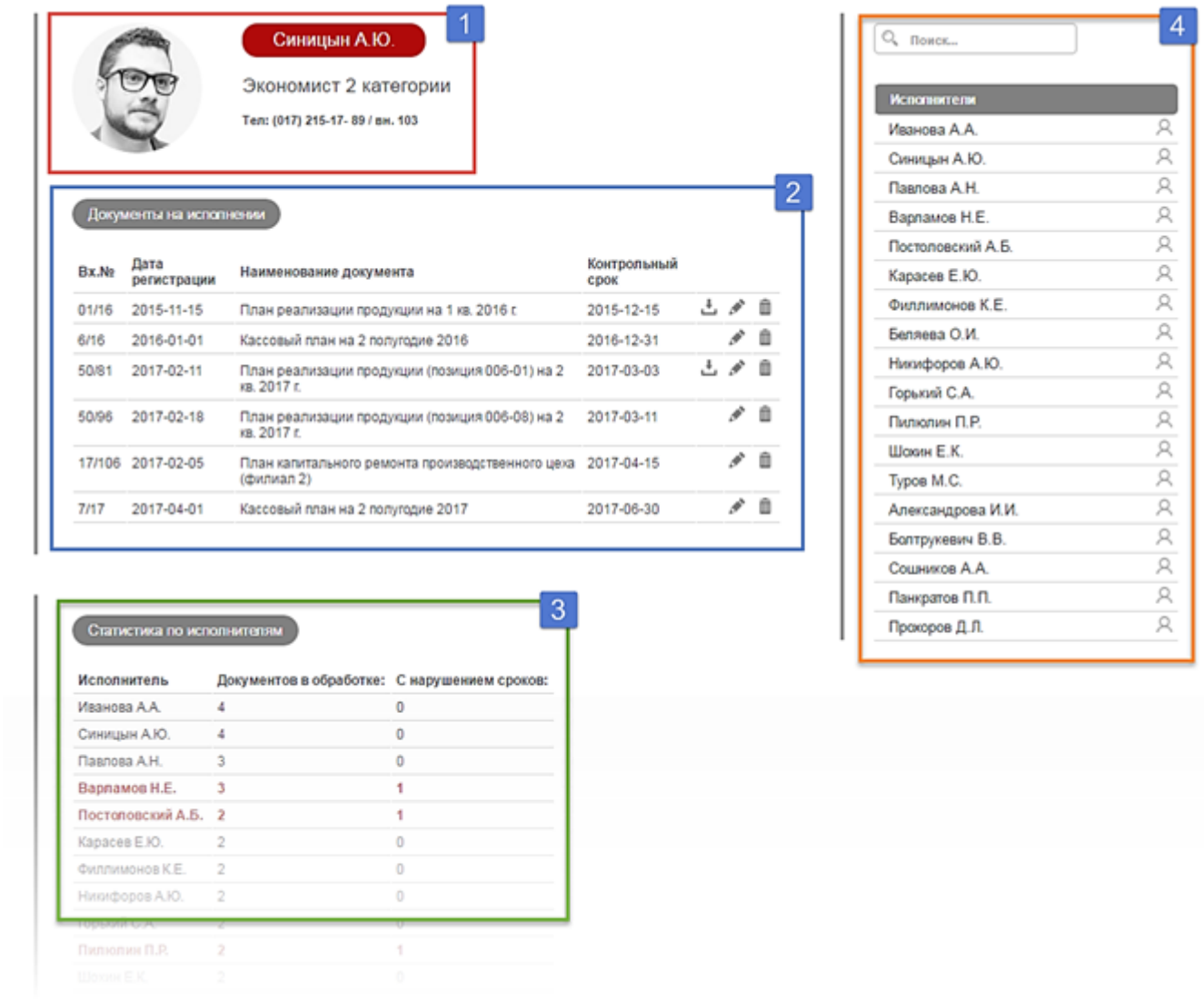


Рисунок 7.4 – Общий вид веб-страницы «Исполнители»

На странице «Сроки» (см. рисунок 7.5) отображены две информационные таблицы, информирующие пользователя о документах с превышением контрольного срока (п.1, рисунок 7.5, область таблицы выделена красным цветом) и документах с истекающим сроком исполнения в ближайшие 10 дней (п.2, рисунок 7.5, область таблицы выделена синим цветом).

Изначально строки таблиц отсортированы по полям «Дней просрочено» и «Осталось дней» соответственно. Однако предусмотрена сортировка по выбранному критерию – для этого пользователю необходимо нажать на любой из заголовков таблицы.

С целью быстрого доступа к странице конкретного исполнителя пользователю необходимо нажать на интересующую его ячейку столбца «Исполнитель».

Для быстрого доступа к определенным действиям с документами пользователю необходимо нажать на соответствующую пиктограмму. Доступны следующие быстрые действия (перечислены в порядке их отображения): «Загрузить с сервера электронный файл документа», «Редактировать документ» и «Удалить документ». Отсутствие пиктограммы «Загрузить с сервера электронный файл документа» означает отсутствие на сервере загруженного файла документа.

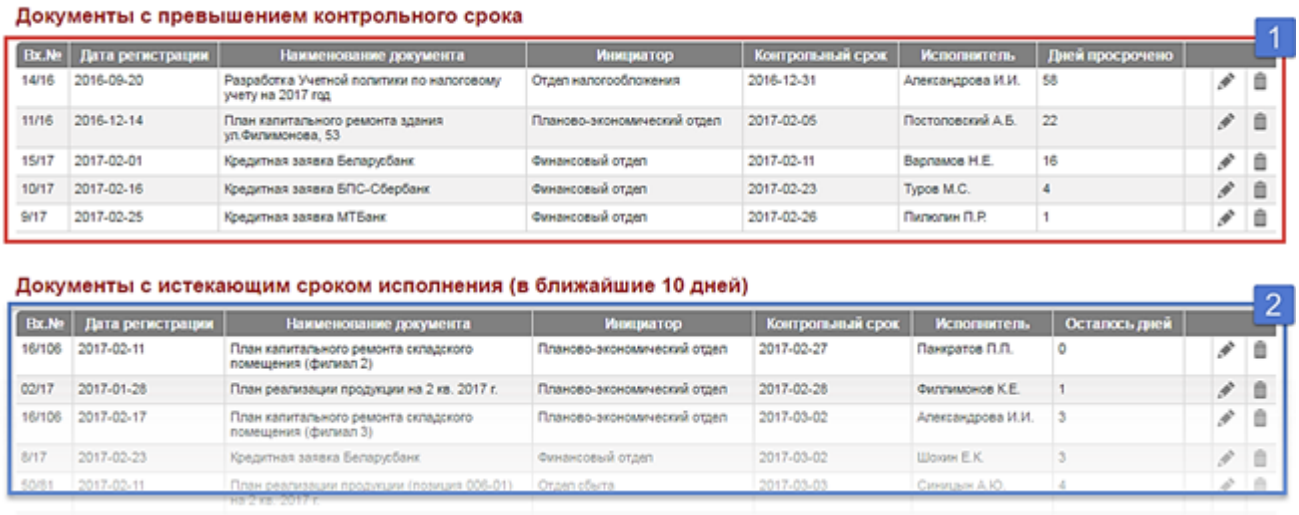


Рисунок 7.5 – Общий вид веб-страницы «Сроки»

Для добавления, изменения и удаления записей о документах в Приложении доступны соответствующие веб-страницы, которые вызываются способами, описанными выше. На рисунке 7.6 представлен общий вид страницы добавления документа.

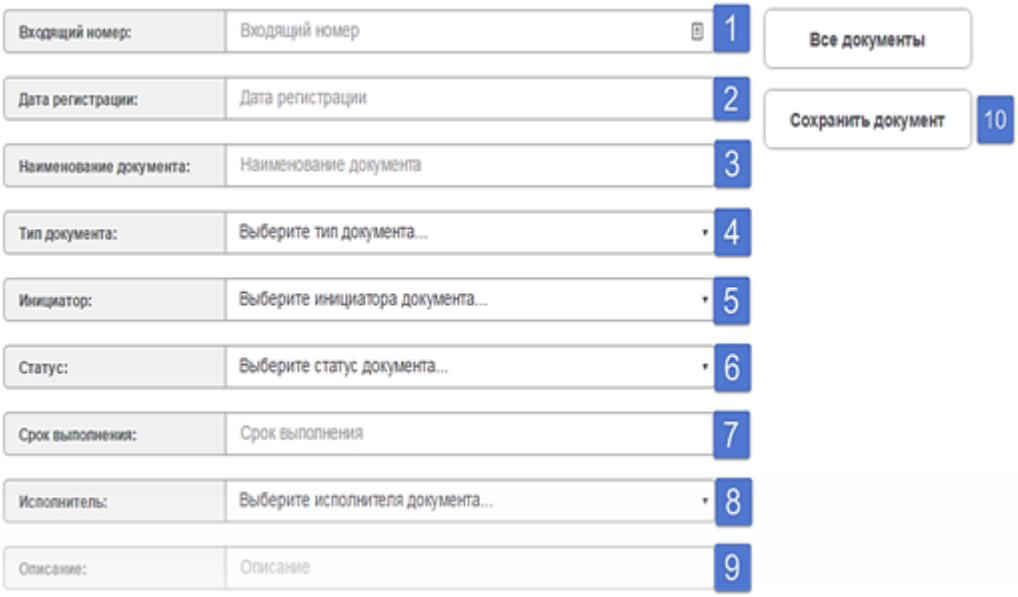


Рисунок 7.7 – Общий вид веб-страницы «Добавление документы»

Данная страница состоит из 9 текстовых полей или полей одиночного выбора, обязательными к заполнению являются поля 1-8 (рисунок 7.6). После заполнения указанных полей с целью сохранения введенной информации о документе в базу данных пользователю необходимо нажать кнопку «Сохранить документ» (п.10, рисунок 7.6), после чего произойдет перенаправление на страницу «Документы», в таблице которой уже будет отображена введенная информация. В случае некорректного ввода произойдет перенаправление на страницу с ошибкой (см. рисунок 7.7), а внесения информации в базу данных не произойдет.

Рисунок 7.7 – Общий вид веб-страницы «Ошибка обработки документов»

При доступе к странице редактирования вновь или ранее созданного документа дополнительно станет доступным поле загрузки файла, если ранее файл загружен не был (см. рисунок 7.8).

Рисунок 7.8 – Вид области добавления файла веб-страницы «Редактирование документа»

К загрузке на сервер доступны файлы размером до 10 Мб. После нажатия на кнопку «Добавить» (п.2, рисунок 7.8) и выбора файла в области справа появится кнопка «Загрузить файл» (п.3, рисунок 7.8), по нажатию которой произойдет загрузка файла на сервер.

8 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СИСТЕМЫ И ОЦЕНКА ВЫПОЛНЕНИЯ ПОСТАВЛЕННЫХ ЗАДАЧ

Для всех методов бизнес-логики были использованы юнит-тесты. В результате их выполнения была получено 100% прохождения тестирования. Это означает, что классы бизнес-логики предусматривают работу во всех возможных ситуациях использования кода.

В программе учтены следующие элементы:

- валидация вводимой информации;
- контроль целостности транзакций.

При тестировании функционирования системы не выявлено никаких сбоев в системе.

Проведена ревизия кода на предмет ошибок синхронизации доступа к общим ресурсам. Были выявлены и устранены ошибки синхронного доступа к некоторым коллекциям и объектам.

Нагрузочное тестирование показало, что в процессе использования приложения до ста пользователями одновременно не происходило никаких сбоев и перегрузок сервера.

При значительном росте данных в базе скорость запросов снижается, но остается на приемлемом уровне.

Таким образом, можно заключить, что поставленные требования к системе выполнены в достаточной мере.

Заключение

Подготовка к разработке приложения «Документация структурного производственного отдела» была начата с ознакомления задачами и функциями подразделения предприятия, а также с процессами и этапами учета документации, поступающей на исполнение в данное подразделение. Полученная информация стала основой для создания веб-приложения с серверной частью с организацией взаимодействия с соответствующей базой данных. В ходе

разработки произведен обзор методов и алгоритмов решения поставленной задачи, осуществлено функциональное моделирование проектируемой системы на основе стандарта IDEF0, составлена информационная модель системы, а также разработаны модели системы на основе стандарта UML 2.0. В ходе реализации серверной части приложения организовано взаимодействие с базой данных MySQL посредством драйвера, предоставленного производителем СУБД. В процессе разработки сервера реализованы функции отправки/получения данных из/в СУБД, их консолидации и формирования веб-страниц, отображаемых в браузере пользователя. С целью освобождения от значительного объёма сравнительно низкоуровневого программирования при работе с реляционной базе данных изучены и применены технологии объектно-реляционного отображения (ORM), такие как Hibernate, и сопутствующий язык запросов HQL. В ходе реализации веб-интерфейса приложения разработана и внедрена единая таблица стилей (CSS) jsp-страниц, применен инструментарий языка JSTL. При помощи языка JavaScript и сопутствующих библиотек (jQuery, sortable и др.) добавлена интерактивность для веб-содержимого, что облегчило взаимодействие пользователя с веб-приложением. Разработанное приложение обеспечивает ведение базы данных входящих документов (добавление, редактирование, удаление информации о документах, поступивших на исполнение в отдел), загрузку на/с сервер(а) электронных версий документов, отображение статистики по документам, находящихся на исполнении у конкретного сотрудника, отображение актуальной информации о количестве документов с нарушением контрольных сроков исполнения, отображение актуальной информации о количестве документов с нарушением контрольных сроков исполнения, находящихся на исполнении у конкретного сотрудника производственного отдела. Исходя из изложенного выше, можно сделать вывод о том, что разработанное приложение может использоваться по назначению в представленной предметной области.

Список использованных источников

1. [печатное издание] Буч Г. / Язык UML. Руководство пользователя. / Грэйди Буч, Джеймс Рамбо, Айвар Джекобсон: Пер. с англ. Слинкин А. А. – 2-ое изд., стер. – М.: ДМК «Пресс»; СПБ.: Питер, 2006 – 432 с.: ил.
2. [печатное издание] Глухова Л.А. / Технологии разработки программного обеспечения. Учеб. пособие.
3. [печатное издание] Методология функционального моделирования IDEF0 / РД IDEF 0 – 2000.
4. [печатное издание] Т. М. Унучек [и др.]. / Языки программирования для разработки сетевых приложений: язык программирования JAVA: лаб. практикум для студ. спец. I-27 01 01 «Экономика и организация производства», I-26 02 03 «Маркетинг» днев. формы обуч. В 2 ч. /– Минск : БГУИ
5. [печатное издание] Блинов И.Н., Романчик В.С. Java. Промышленное программирование – УниверсалПресс, 2007. - 704с.
6. [печатное издание] **Методология функционального моделирования IDEF0 / РД IDEF 0 – 2000.**
7. [печатное издание] **Фаулер М. / UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2007. – 192 с., ил.**
8. [печатное издание] **Леоненков А. / Самоучитель UML, 2-е издание. – СПб: БХВ-Петербург, 2005. – 432 с.: ил.**
9. [печатное издание] **Монахов В.В. / Язык программирования и среда NetBeans – 3-е издание перераб. И доп. –СПб.: БХВ-Петербург, 2011. –704 с.**
10. [печатное издание] **UML. Классика CS. 2-у изд./Пер. с англ.; Под общей редакцией проф. С.Орлова - СПб.: Питер, 2007. - 736 с.: ил.**
11. [печатное издание] **Басс Л., Клементс П., Кацман Р. Архитектура программного обеспечения на практике. 2-е издание. - СПб.: Питер, 2007. - 575 с.: ил.**

12. [url] **github** https://github.com/AnnaAntonova123/KP

Приложения

1. [Приложение] **Приложение** 5ed40a37621e2_приложение.docx
2. [электронный документ] 5ed40ec4b850d_лист задания.docx
3. [Задание] **Задание** 5ed40ede39271_лист задания.docx