Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Отчет

по лабораторной работе №5

на тему:

**Проектирование пользовательского интерфейса. Архитектурные решения. Описание и разработка алгоритмов, реализующих бизнес-логику программного средства**

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО преподавателя

(подпись)

Выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО студента

(подпись)

Минск, 2021

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

Одним из требований к хорошему графическому интерфейсу программной системы является концепция «предсказуемости», чтобы система работала предсказуемо, чтобы пользователь заранее интуитивно понимал, какое действие выполнит программа после получения его команды.

Существует три основных критерия качества пользовательского интерфейса:

– скорость работы пользователей;

– количество человеческих ошибок;

– скорость обучения.

Рассмотрим каждый из них. Первый – скорость работы пользователя. Согласно Дональду Норману, взаимодействие пользователя с системой (не только компьютерной) состоит из семи шагов:

– формирование цели действий;

– определение общей направленности действий;

– определение конкретных действий;

– выполнение действий;

– восприятие нового состояния системы;

– интерпретация состояния системы;

– оценка результата.

Таким образом, процесс размышления занимает почти все время, в течение которого пользователь работает с компьютером, т. к. шесть из семи этапов полностью заняты умственной деятельностью. Соответственно, повышение скорости этих размышлений приводит к существенному улучшению скорости работы.

Существенно повысить скорость собственно мышления пользователей невозможно, но качественный пользовательский интерфейс должен уменьшить влияние факторов, усложняющих (и, соответственно, замедляющих) процесс мышления.

Второй – количество человеческих ошибок. Пользовательский интерфейс должен содержать элементы, которые позволят уменьшить количество допускаемых ошибок. К этим элементам относятся:

– плавное обучение пользователей в процессе работы;

– снижение требований к бдительности;

– повышение разборчивости и заметности индикаторов.

Кроме того, пользовательский интерфейс должен содержать средства, позволяющие снизить чувствительность системы к ошибкам. К ним относятся:

– блокировка потенциально опасных действий пользователя до получения подтверждения правильности действия;

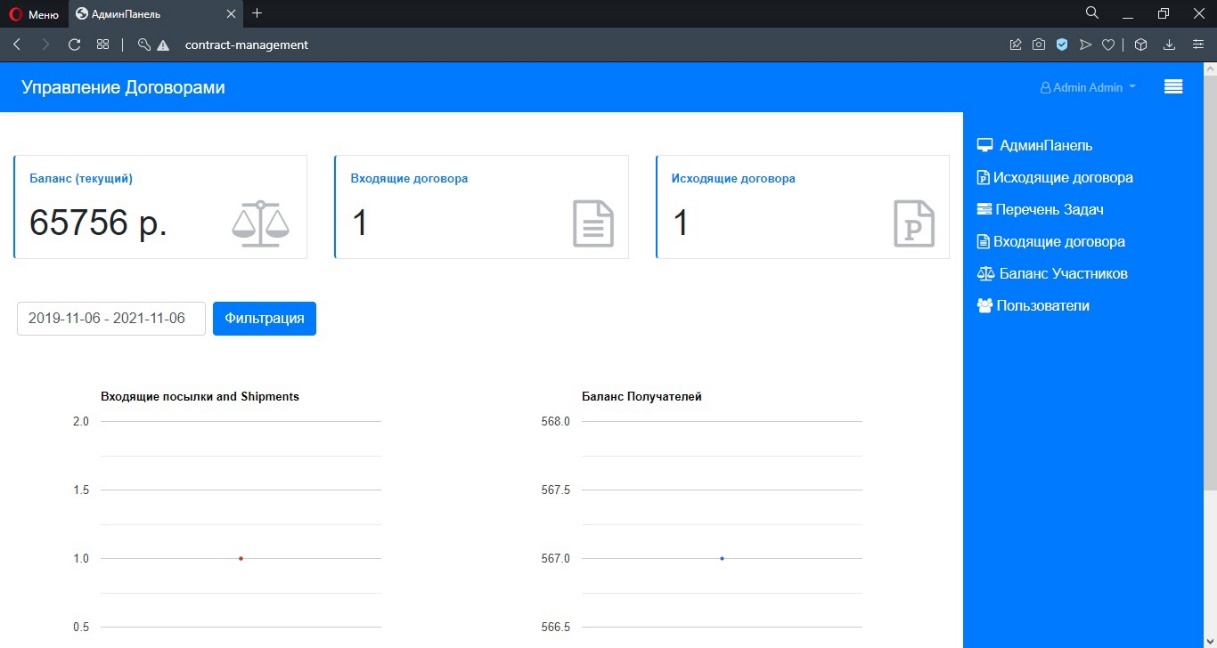
– проверка системой всех действий пользователя перед их принятием;

– самостоятельный выбор системой необходимых команд или параметров, когда от пользователя требуется только проверка.

И последний – это скорость обучения. Пользовательский интерфейс должен содержать средства, позволяющие пользователю в максимально короткие сроки научиться работать с программой или системой. К таким средствам относятся различного вида справочные системы, подсказки, информационные сообщения.

Все эти три критерия качества пользовательского интерфейса были учтены при проектировании программного средства автоматизации договорной деятельности торгового предприятия.

Теперь представим пользовательский интерфейс:



Теперь перейдем к рассмотрению архитектурных решений.

Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

В программной инженерии шаблон проектирования приложений – это многократно применяемое решение регулярно возникающей проблемы в рамках определённого контекста архитектуры приложения. Шаблон – это не законченное архитектурное решение, которое можно напрямую преобразовать в исходный или машинный код. Это описание подхода к решению проблемы, который можно применять в разных ситуациях.

Данное программное средство будет разрабатываться на фреймворке Laravel.

Laravel – бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC (англ. Model View Controller – модель-представление-контроллер). Исходный код проекта размещается на GitHub.

Laravel – это удобный и популярный php-фреймворк, предназначенный для разработки сайтов. Обладает открытым исходным кодом, что позволяет его дорабатывать, адаптируя под любые проекты. Laravel-фреймворк основан на компонентах Symfony, поэтому в нем применяется кодовая база высокого качества.

Разработка сайтов на Laravel подходит для проектов любого уровня сложности, среди которых могут быть стартапы, корпоративные ресурсы, интернет-магазины, маркетплейсы, социальные сети. Фреймворк позволяет делать букинг-сервера, аукционы, мобильные приложения, сервисы на основе набора REST API. Здесь создаются ресурсы с логичной архитектурой и упорядоченной структурой, а также порталы с неограниченным набором функций.

Преимущества создания веб-приложений с использованием фреймворка Laravel:

1 Laravel-фреймворк обновляется примерно раз в полгода. Это служит гарантией безопасности его использования и увеличивает производительность движка. Laravel не нагружает сервер.

2 Данный фреймворк заслужил немалую популярность и собрал вокруг себя большое сообщество, что позволяет обмениваться навыками, а также решать вопросы любого уровня сложности. Совместное использование открыло доступ к бесплатному скачиванию базовых модулей, которые идентичны для большинства проектов.

3 Процесс установки и настройки Laravel очень прост. Фреймворк поддерживает разные версии ресурса. Разработчики могут тестировать обновления в дополнительной версии, а затем вносить изменения в основную. Встроенные во фреймворк инструменты защищают от XSS и SQL-инъекций.

Думаю, что на этом можно завершить обоснование выбора основного фреймворка для разработки данного программного средства.

Следующий фреймворк, который было решено использовать – это Bootstrap.

Bootstrap (также известен как Twitter Bootstrap) – свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Bootstrap – это открытый и бесплатный HTML, CSS и JavaScript фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений.

Фреймворк Bootstrap используется по всему миру не только независимыми разработчиками, но иногда и целыми компаниями.

Основная область его применения – это фронтенд разработка сайтов и интерфейсов админок. Среди аналогичных систем (Foundation, UIkit, Semantic UI, InK и других) фреймворк Bootstrap является самым популярным.

Фреймворк Bootstrap позволяет верстать сайты в несколько раз быстрее, чем это можно выполнить на «чистом» CSS и JavaScript. А в нашем мире время – это очень ценный ресурс. Кроме этого, его популярность ещё обусловлена доступностью. Она заключается в том, что на нём даже начинающий разработчик может верстать достаточно качественные макеты, которые трудно было бы выполнить без глубоких знаний веб-технологий и достаточной практики.

Преимущества Bootstrap при его использовании для frontend разработки:

– высокая скорость создания качественной адаптивной вёрстки даже начинающими веб-разработчиками (достигается это благодаря использованию готовых классов и компонентов, созданных профессионалами);

– кроссбраузерность и кроссплатформеннось (корректное отображение и работа сайта во всех поддерживаемых этим фреймворком браузерах и операционных системах);

– наличие большого количество готовых хорошо продуманных компонентов, протестированных огромным сообществом веб-разработчиков на различных устройствах;

– возможность настройки под свой проект, достигается это посредством изменения SCSS переменных и использования миксинов (можно изменить количество колонок, цвета, радиус скруглений, отступы между колонками и так далее);

– низкий порог вхождения; для работы с фреймворком не обязательно иметь «глубокие» знания по HTML, CSS, JavaScript и jQuery (достаточно знать только основы этих технологий);

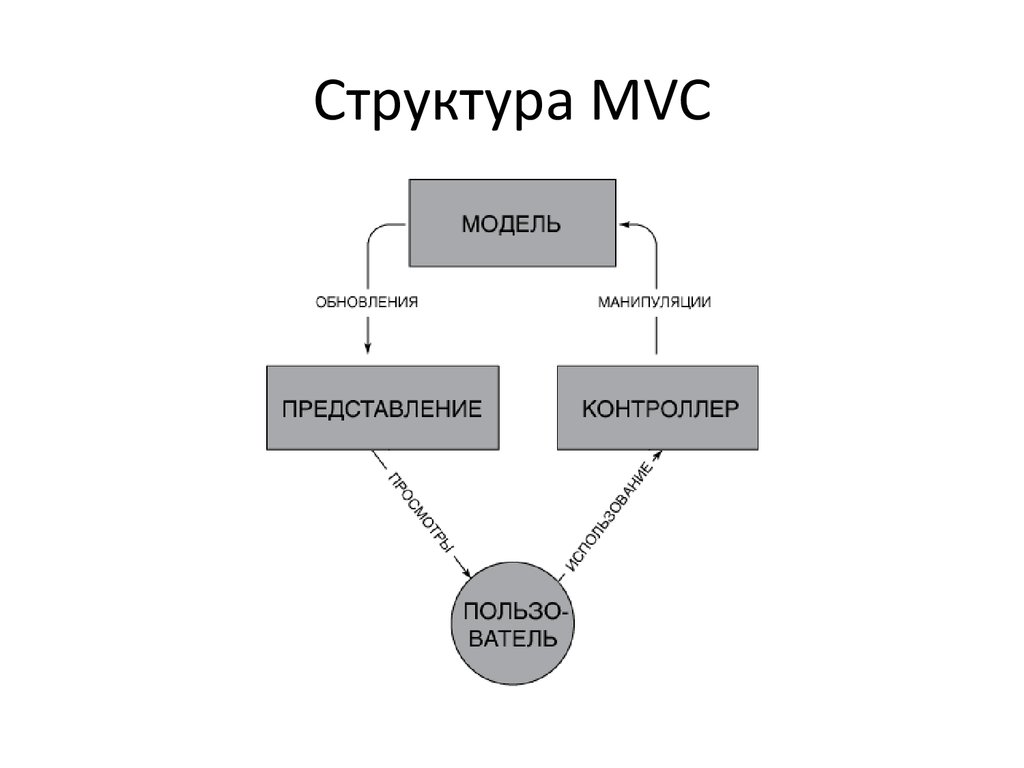
– однородность дизайна и его согласованность между различными компонентами (в Bootstrap все компоненты выполнены в едином стиле);

– наличие огромных сообществ и обучающих материалов; при желании это поможет не только хорошо разобраться в фреймворке, но и найти ответы практически на любые вопросы.

Думаю, что на этом можно завершить обоснование выбора фреймворка Bootstrap для разработки данного программного средства.

И последнее, что необходимо описать в данном подразделе это шаблон проектирования MVC. Как было сказано выше, на данном паттерне основан фреймворк Laravel.

Концепция MVC (Model – View – Controller) – схема использования нескольких паттернов 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. Структура паттерна MVC:



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

1 Модель отвечает за управление данными, она сохраняет и извлекает сущности, используемые приложением, как правило, из базы данных и содержит логику, реализованную в приложении.

2 Представление несет ответственность за отображение данных, которые даёт контроллер. С представлением тесно связано понятие шаблона, который позволяет менять внешний вид показываемой информации. В веб-приложении представление часто реализуется в виде HTML-страницы.

3 Контроллер связывает модель и представление. Он получает запрос от клиента, анализирует его параметры и обращается к модели для выполнения операций над данными запроса. От модели поступают уже скомпонованные объекты. Затем они перенаправляются в представление, которое передаёт сформированную страницу контроллеру, а он, в свою очередь, отправляет её клиенту.

Шаблоны проектирования, такие как MVC, определяются, прежде всего, чтобы облегчить общение разработчиков между собой. Вместо того чтобы говорить коллеге: «Давай введем абстракцию доступа к данным, потом организуем отдельный уровень обработки вывода данных, а между ними вставим посредника», вы можете обратиться к общепринятому лексикону и сказать: «А давай-ка применим здесь шаблон проектирования MVC».

Основная цель применения этой концепции состоит в отделении бизнес-логики от её представления. За счет такого разделения повышается возможность повторного использования. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. При этом, выполняются следующие задачи:

1 К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы.

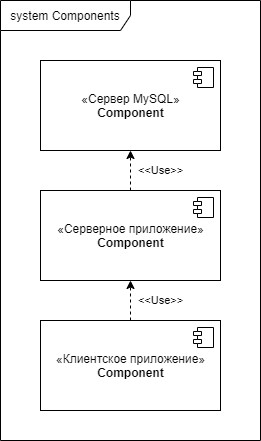
2 Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных), для этого достаточно использовать другой контроллер.

3 Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

Важно отметить, что как представление, так и контроллер зависят от модели. Однако модель не зависит ни от представления, ни от контроллера. Тем самым достигается назначение такого разделения: оно позволяет строить модель независимо от визуального представления, а также создавать несколько различных представлений для одной модели.

Также архитектуру данного программного средства можно рассмотреть на примере диаграммы компонентов и диаграммы классов.

Описание компонентов системы предполагает создание диаграммы компонентов. Данная диаграмма:

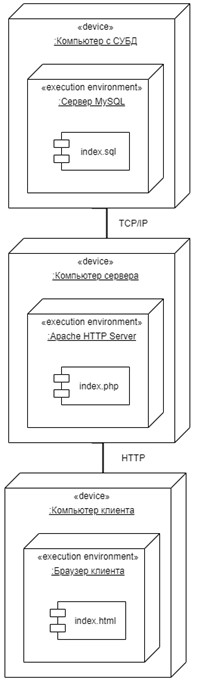


Программное средство имеет трехуровневую архитектуру, то есть состоит из трёх основных компонентов:

1. «Клиентское приложение» – это компонент, который выполняет построение графического пользовательского интерфейса и управление им. Использует внешние интерфейсы компонента «Серверное приложение».
2. «Серверное приложение» – это компонент, который осуществляет обработку данных, их валидацию, содержит бизнес-логику. Использует внешние интерфейсы компонента «Сервер MySQL».
3. «Сервер MySQL» – это компонент, который осуществляет работу с данными, отвечает за их хранение, позволяет их получать.

При этом, каждый из компонентов предполагается размещать на разных персональных компьютерах (или телефонах). Естественно, разместить можно и на одном девайсе.

Для описания размещения системы была создана диаграмма развёртывания. Диаграмма развертывания:



На диаграмме представлены три узла:

1. «Компьютер клиент» – это устройство пользователя системы, на котором выполняется такой компонент, как «Клиентское приложения». При этом, внутри данного узла размещается узел среды выполнения «Браузер клиента». Это может быть любой современный браузер, через который клиент работает. В среде выполнения выполняется файл «index.html», который отвечает за разметку страницы и подгружает исполняемые скрипты и стили. Обменивается с узлом «Компьютер сервера» данными посредствам протокола HTTP (в перспективе – HTTPS).
2. «Компьютер сервера» – это устройство на сервере, на котором выполняется такой компонент, как «Серверное приложения». При этом, внутри данного узла размещается узел среды выполнения «Apache HTTP Server». В среде выполнения выполняется файл «index.php», который является точной входа в проект. Обменивается с узлом «Компьютер с СУБД» данными посредствам протокола TCP/IP.
3. «Компьютер с СУБД» – это устройство, на котором выполняется такой компонент, как «Сервер MySQL». При этом, внутри данного узла размещается узел среды выполнения «Сервер MySQL». В среде выполнения выполняется файл «index.sql», который является базой данных приложения.

Описанные выше компоненты и узлы позволяют организовать работу современного веб-приложения с разграничением функционала и распределением нагрузки.

Далее следует рассмотреть диаграмму классов.

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Диаграмма классов:

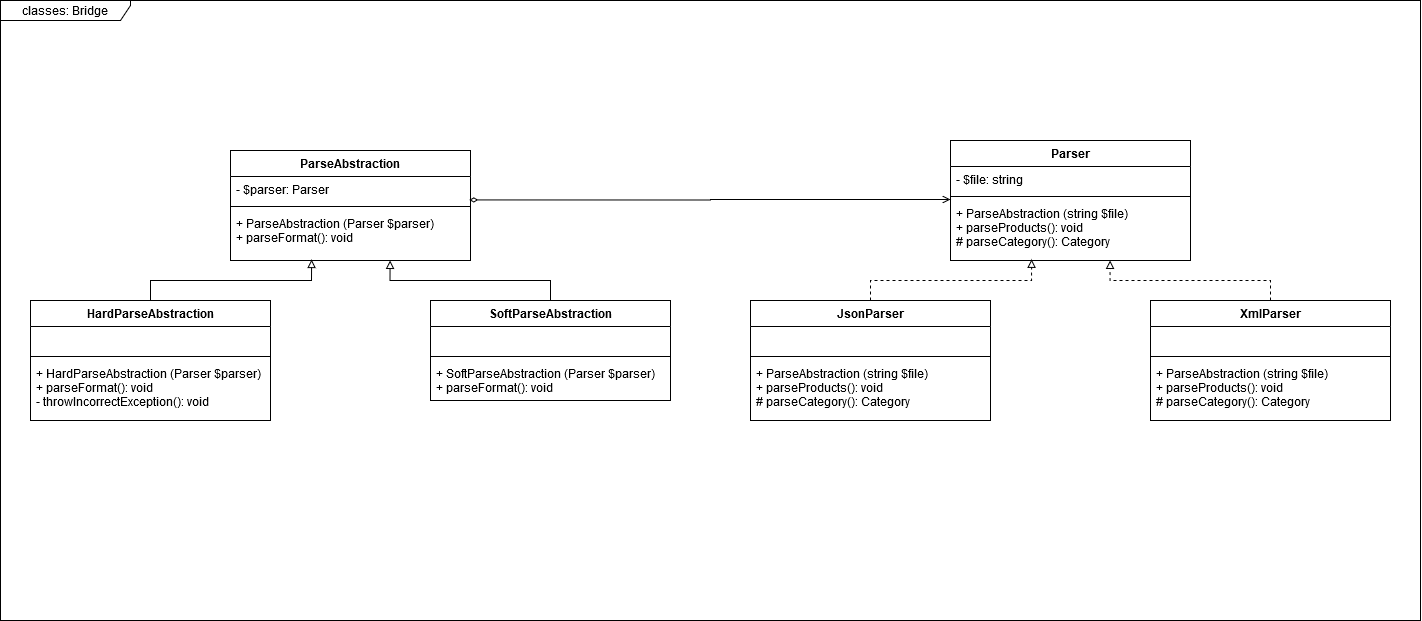


Диаграмма классов является ключевым элементом в объектно-ориентированном моделировании. На диаграмме классов представлены в рамках три компонента:

1 В верхней части написано имя класса. Имя класса выравнивается по центру и пишется полужирным шрифтом. Имена классов начинаются с заглавной буквы. Если класс абстрактный – то его имя пишется полужирным курсивом.

2 Посередине располагаются поля (атрибуты) класса. Они выровнены по левому краю и начинаются с маленькой буквы.

3 Нижняя часть содержит методы класса. Они также выровнены по левому краю.

Рассмотрим алгоритм фильтрации данных в таблице по введенному параметру. Данный алгоритм:



Для фильтрации данных необходимо ввести параметр фильтрации, если параметр не будет введен система запросит его ввести. Следующий этап – это считывание параметра с поля и отправка запроса на сервер. После выполнения данной операции система будет ждать ответ. Когда ответ будет получен, данные в таблице отфильтруются.