ВВЕДЕНИЕ

Программное средство учета и контроля студентов в обжещитии является сервисом, позволяющим лицам, ответственным за пребывание студентов в общежитии, вести строгий мониторинг заселения(выселения) студентов, получать информацию о комнатах и блоках, в которых проживают учащиеся. Помимо этого предоставлена возможность вести учет выговоров и дисциплинарных взысканий, полученных студентами во время проживания в

общижитии, что является существенным показателем при рассмотрении перезаселения студента на предстоящий учебный год.

Ежегодно иногородние студенты, не имеющие постоянного жилья, заселяются в общежитие, предоставляемое ВУЗом. Процесс заселения в общежитие довольно продолжителен, контроль над ним осуществляется со стороны заведующей, коменданта и пр. Так как документооборот, непосредственно связанный с этим процессом, не оцифрован и не оптимизирован, возникают некоторые трудности при определении свободных комнат для проживания. Это происходит по ряду причин : не учтены студенты, выселившиеся из общежития по окончанию ВУЗа; не учтены студенты, имеющие льготы и первоочередное право заселения в общежитие; не учтены студенты, имеющие дисциплинарные взыскания или выговоры и пр.

Преимуществом данного программного средства является снижение возможных просчетов при заселении(выселении) студентов, определении свободных помещений для проживания. Просмотр имеющейся информации о студенте позволит проанализировать длительность его проживания в общежитии, учитывать его дисциплинарные взыскания или их отсутствие для составления плана заселения общежития на будущий учебный год или полугодие.

Целью данного дипломного проекта является разработка и реализация программного средства учета и контроля студентов в общежитии, которое обеспечит полный контроль пребывания учащихся в общежитии ВУЗа и позволит проводить необходимую аналитику при прогнозировании их будущего перезаселения(выселения). В первую очередь данная система предназначена для лиц, ответственных за проживание студентов в общежитии (комендант, заведующая, воспитатель и тд.). Также система может использоваться работниками университета, занимающиеся сбором информации о студентах.

В соответствии с поставленной целью были определены следующие задачи:

1. Выбор платформы для создания системы.
2. Разработка пользовательской и административной частей системы.
3. Учет комнат(блоков), заселенных студентами.
4. Учет выговоров и дисциплинарных взысканий студентов.
5. Построение системы сбора данных для статистики и аналитики.
6. Система будет состоять из веб- приложения, которое будет предоставлять следующие функции:

Функции коменданта:

1. Добавление общехозяйственных часов студенту.
2. Снятие общехозяйственных часов со студента.
3. Просмотр информации по выбранному студенту.
4. Поиск учетной записи студента.
5. Проставление оценок блокам и комнатам по итогам отчетного периода.

Функции заведующей:

1. Добавление учетной записи студента в системе.
2. Просмотр заселенных комнат по этажам.
3. Просмотр заселенных комнат всего общежития.
4. Удаление учетной записи студента из системы.
5. Редактирование учетной записи студента.
6. Поиск учетной записи студента.
7. Добавлениедисциплинарного взыскания студенту.
8. Снятие дисциплинарного взыскания со студента.
9. Редактирование дисциплинарного взыскания студента.
10. Просмотр общехозяйственных часов, выполненных студентом.
11. Просмотр оценок комнат за отчетный период.

**1** ОБЗОР ЛИТЕРАТУРЫ

К моменту появления понятия веб-сервиса [1] уже существовали технологии, с помощью которых приложения могли взаимодействовать на расстоянии, где одна программа могла вызвать какой-нибудь метод в другой программе, которая при этом могла быть запущена на компьютере, который в свою очередь мог располагаться в другом городе или даже стране. Все этого сокращенно называется RPC (Remote Procedure Calling – удаленный вызов процедур). В качестве примеров можно привести технологии CORBA, а для Java – RMI (Remote Method Invoking – удаленный вызов методов). И все вроде в них хорошо, особенно в CORBA, т.к. с ней можно работать на любом языке программирования, но чего-то все же не хватало. Было решено разработать стандарт, который должен был отражать основную идею веб-сервиса, а именно, создание такого RPC, который будет засовываться в http пакеты. Был разработан общий подход, позволяющий сделать такое.

В веб-сервисах всегда есть клиент и сервер. Если посмотреть на веб-сервисы в разрезе стека сетевых протококолов, мы увидим, что это, в классическом случае, не что иное, как еще одна надстройка поверх протокола HTTP [2]. По сути, веб-сервисы представляют собой реализацию абсолютно четких интерфейсов, предполагающих обмен данными между приложениями, которые могут быть написаны не только на разных языках, но и быть распределенными на разных узлах сети.

На сегодняшний день наиболее распространены следующие протоколы реализации веб-сервисов: Simple Object Access Protocol, Representational State Transfer, XML Remote Procedure Call. Если вдаваться в подробности, то SOAP является следующий ступенью развития XML-RPC, в то время как концепция REST [3] предполагает архитектурный стиль, нежели новую технологию, в основе которой лежит теория CRUD (Create Read Update Delete) в контексте концепций WWW. Безусловно, круг протоколов не ограничивается только ними, но так остальные не получили должного распространения, рассматривать их не имеет смысла.

SOAP более применим в сложных архитектурах, где взаимодействие с объектами выходит за рамки теории CRUD, а вот в тех приложениях, которые не покидают рамки данной теории, вполне применимым может оказаться именно REST ввиду своей простоты и прозрачности. Помимо этого REST может дать выигрыш в производительности, так как он не требует разбор сложных xml команд на сервере(выполняются простые http-запросы).

Основным языком разработки web-сервиса является Java. В течение последних нескольких лет разработчики создавали REST-сервисы для своих Java-приложений, используя самые разнообразные технологии. Архитектура REST отличается своей простотой, требуя от приложений обеспечить только возможность приема сообщений с HTTP-заголовками. Эта функция легко реализуется простыми Web-контейнерами для Java-приложений.

REST-приложения часто создаются на основе сервлетов. Сервлеты не предписывают какие-либо конкретные походы к разработке. Как правило, сервлеты получают на обработку запросы, анализируют их заголовки, в том числе URI, чтобы определить, к какому ресурсу выполняется обращение. Ряд API был создан на основе этой простой модели сервлетов. Несмотря на все усилия по формализации, ни один из этих API не превратился в официальный стандарт.

# Язык программирования Java

На сегодняшний день язык программирования Java [4] является одним из наиболее распространенных и популярных. Первая версия языка появилась в 1996 году в компании Sun Microsystems, которая была поглощена компанией Oracle. Язык программирования Java задумывался как универсальный, который быо бы применим для различного рода задач. И к настоящему мементу Java проделал большой путь, было издано множество версий, текущей является Java 8. По прошествии времени Java превратился из просто универсального языка в целую платформу, которая объединила различные технологии, используемые для различных типов задач: от создания десктопных приложений до написания крупных веб-порталов. Кроме того, язык программирования Java активно применяется для создания софта для целого ряда устройств: обычных ПК, планшетов, смартфонов, бытовой техники.

Ключевой особенностью языка программирования Java является то, что его код сначала транслируется в специальный байт-код, который является независимым от платформы, затем он выполняется Java Virtual Machine. В этом плане Java отличается от стандартных интерпретируемых языков как например PHP или Perl, код которых сразу же выполняется интерпретатором. В то же время Java не является и чисто компилируемым языком, как С или С++.

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Solaris и т.д. Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код.

Еще одной ключевой особенностью Java является то, что она поддерживает автоматическую сборку мусора. А это значит, что вам не надо освобождать вручную память от ранее использовавшихся объектов, как в С++, так как сборщик мусора это сделает автоматически за вас.

Java является объектно-ориентированным языком. Он поддерживает полиморфизм, наследование, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в то же время гибких, масштабируемых и расширяемых приложений, что, безусловно, является одним из наиболее значимых его плюсов.

# Обзор web-фреймворков

Несомненно, Java сейчас является одним из самых популярных языков, для него существует множество подключаемых инструментов. Приведу в пример несколько из них, после чего будет дан ответ, почему для разработки был выбран тот или иной.

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Spring Framework предоставляет бо́льшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.Между тем, особенности ядра Spring Framework применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.Spring Framework может быть рассмотрен как коллекция меньших фреймворков или фреймворков во фреймворке. Большинство этих фреймворков может работать независимо друг от друга, однако они обеспечивают большую функциональность при совместном их использовании.

[Vaadin](https://vaadin.com/home) базируется на [Google Web Toolkit](http://www.gwtproject.org/) (он используется для вывода элементов пользовательского интерфейса и взаимодействия с сервером на стороне клиента), что добавляет архитектуре этого фреймворка сложности. Тем не менее, знание Google Web Toolkit не является обязательным для разработки на Vaadin. Более того, этот фреймворк вообще один из самых простых для изучения, а главной его «фичей» является «режим дизайна» —  в нем можно строить пользовательский интерфейс в режиме WYSIWIG и писать логику отдельно для каждого компонента. Vaadin идеально подходит как для новичков, так и для профессионалов. Из минусов: не самая развитая экосистема и плохая масштабируемость для больших сайтов.

[JSF](http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html) является частью [Java EE](http://www.oracle.com/technetwork/java/javaee/overview/index.html) и официально поддерживается Oracle. Хоть этот фреймворк и не очень подходит для быстрой разработки, он легок в использовании благодаря наличию отличной документации (еще бы, ведь ее предоставляет сама Oracle), отсутствию каких-либо внешних зависимостей (пока вы остаетесь в экосистеме Java EE) и богатству возможностей. Экосистема фреймворка крайне развита и представляет из себя множество библиотек на все случаи жизни, в том числе и инструменты для удобной разработки пользовательского интерфейса. Главная фишка JSF в том, что, как уже было сказано ранее, он является частью Java EE — из этого следует отличное взаимодействие с IDE и официальная поддержка от Oracle. Все это в совокупности значительно облегчает разработку.  Недостаток фреймворка в сложности его устройства. Впрочем, иначе быть и не может, ведь он изначально задумывался как единое целое с Java EE.

Следующие фреймворки отличаются от вышеприведенных тем, что они базируются не на Java, а на языках, продолжающих ее традиции и работающих на JVM. [Play](https://www.playframework.com/) написан на [Scala](http://www.scala-lang.org/) и Java (при разработке можно использовать оба языка, но изначально фреймворк предназначен для Scala). Одно из главных преимуществ Play — невероятно развитая экосистема, включающая в себя такие вещи, как [SBT](http://www.scala-sbt.org/) (Simple Building Tool) и [Akka](http://akka.io/) (отличная библиотека для реализации реакционного программирования). TypeSafe предоставляет полную документацию по возможностям фреймворка. Кроме того, данный фреймворк отлично подходит для быстрой разработки благодаря своей простоте. Таким образом, Play является отличным выбором для разработчика на Scala и неплохим для Java-программиста (знание первого все-таки желательно, потому что Play переносит во второй язык идиомы из первого, а функциональный код в объектно-ориентированной Java смотрится не к месту).

[Grails](https://grails.org/) — фреймворк, написанный на скриптовом языке [Groovy](https://ru.wikipedia.org/wiki/Groovy), созданный под влиянием Ruby on Rails. Использовать данный фреймворк можно как c Java, так и c Groovy, но последний вариант все-таки предпочтительнее, так как при написании кода на Java вы не сможете использовать большинство замечательных возможностей Grails. Этот фреймворк идеален для быстрой разработки приложений — он не требует долгой настройки и затачивания под задачу. [Документацию](https://grails.org/wiki/Documentation) также стоит отметить — это множество официальных руководств, демонстрационных примеров, видео и т.д. [Экосистема](https://grails.org/ecosystem.html) представляет из себя более 900 плагинов на все случаи жизни.

В результате было принято решение использовать Spring и вот почему.

Основное преимущество Spring'а - возможность разработки приложения как набора слабосвязанных (loose-coupled) компонентов. Чем меньше компоненты приложения знают друг о друге, тем проще разрабатывать новый и поддерживать существующий функционал приложения. Классический пример - управление транзакциями. Spring позволяет вам управлять транзакциями совершенно независимо от основной логики взаимодействия с БД. Изменение этой логики не порушит транзакционность, равно как изменение логики управления транзакциями не сломает логику программы. Spring поощряет модульность. Компоненты можно добавлять и удалять (почти) независимо друг от друга. В принципе, приложение можно разработать таким образом, что оно даже не будет знать, что управляется Spring'ом. Также Spring заметно упрощает модульное тестирование (unit-testing): в компонент, разработанный для работы в IoC контейнере очень легко инжектировать фейковые зависимости и проверить работу только этого компонента. Ну, и в качестве приятного дополнения, Spring сильно облегчает инициализацию и настройку компонентов приложения, позволяя гибко настраивать приложение без существенных изменений Java-кода.

# **1.3** Выбор СУБД

Система базы данных состоит из базы данных и программного обеспечения общего назначения, которая называется системой управления базой данных (СУБД), и служит для управления базой данных; соответствующего оборудования и людей.

Каждая СУБД должна удовлетворять следующим требованиям [6]:

1. Обеспечивать пользователю возможность создавать новые БД и определять их схему (логическую структуру данных) с помощью специального языка – языка определения данных, поддерживать разнообразные представления одних и тех же данных.
2. Позволять «запрашивать» данные и изменять их с помощью языка запросов, или языка манипулирования данными; допускать интеграцию и совместное использование данных различными приложениями.
3. Поддерживать хранение очень больших массивов данных, измеряемых гигабайтами и более, в течение долгого времени, защищая их от случайной порчи и неавторизованного использования, а также — обеспечивать модификацию БД и доступ к данным путем запросов, т.е. гарантировать безопасность и целостность данных.
4. Контролировать доступ к данным одновременно для многих пользователей; исключать влияние запроса одного пользователя на запрос другого пользователя и не допускать одновременный доступ, который может испортить данные, т.е. гарантировать управление параллельным доступом к данным.

Существует множество различных СУБД, удовлетворяющих этим требованиям. Одной из них является MySQL и она обладает большим рядом преимуществ.

Основные преимущества СУБД MySQL:

1. MySQL — свободная реляционная система управления базами данных [7]. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.
2. MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.
3. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

# **1.4** Обзор аналогов

Сервис «Student Hostel». Сервер запускается в консольном режиме и отображает диагностические сообщения о своей работе(подключение и отключение клиентов, выборка и изменение данных таблиц, сообщения об ошибках).

На рис. 1.1 видны возможности, которыми обладает клиент. Функциональные возможности приложения обеспечиваются двумя списками и выпадающим списком:

1. Выпадающий список содержит список этажей общежития. При изменении этого списка происходит обновление списка комнат – отображаются лишь те комнаты, которые находятся на выбранном этаже.

2. Список комнат содержит информацию о комнатах на выбранном пользователем этаже. При щелчке по строке данного списка в нижней части окна клиента выводится список студентов, проживающих в выбранной комнате. Данная опция позволяет просматривать нужную информацию о количестве доступных мест для студентов, а также стоимость проживания в общежитии.

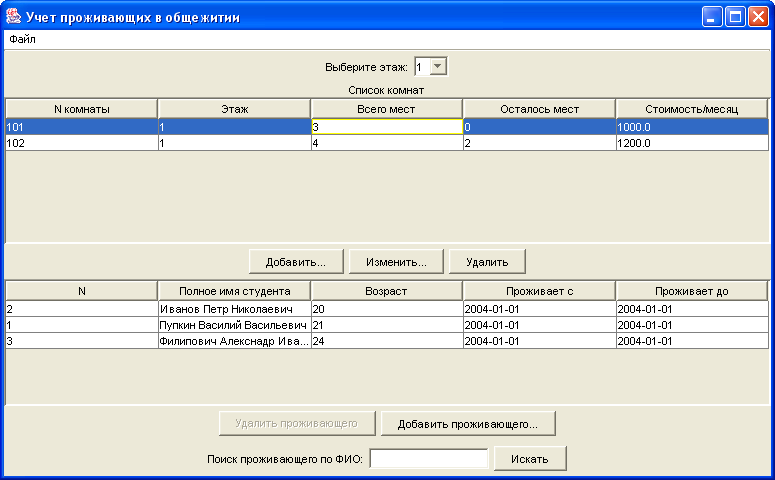


Рисунок 1.1 – Функциональные возможности клиента

Для добавления комнаты необходимо нажать кнопку «Добавить…». Откроется диалог добавления комнаты:

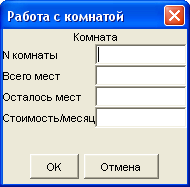


Рисунок 1.2 – Диалог добавления комнаты

Аналогичный диалог, только содержащий уже заполненные поля, появляется при изменении атрибутов комнаты. Для этого необходимо выделить нужную строку в списке комнат и нажать кнопку «Изменить…».

Для добавления проживающих в комнате студентов необходимо выбрать строку с информацией о соответствующей комнате и нажать кнопку «Добавить проживающего…».

Будет открыт диалог добавления сведений о проживающем студенте и работы со списком студентов. Данный диалог позволяет изменять, добавлять и удалять записи о проживающих в общежитии студентах.

Также существует возможность поиска записей по подстроке ФИО студента и их сортировки по ФИО студента – как по убыванию, так и по возрастанию. Для изменения записи необходимо выделить требуемую строку в списке студентов – данные о студенте будут занесены в элементы управления диалога и станут доступными кнопки редактирования/удаления записи.

Для добавления выбранного студента в список проживающих в комнате, выбранной на основном окне клиента, необходимо выделить в списке студентов требуемого студента, выбрать даты заселения и ожидаемого выселения студента, после чего нажать кнопку «Добавить сведения о проживании» - данные будут внесены в таблицу и диалоговое окно будет закрыто.

Для удаления сведений о проживании студента необходимо выбрать требуемую комнату, студента, после чего нажать кнопку «Удалить проживающего». Сведения о проживающих в комнате студентах будут обновлены.

Для поиска сведений о проживании в общежитии студентов с определенными ФИО используется поле редактирования, именованное «Поиск проживающего по ФИО», а также кнопка «Искать». Сведения о студентах при поиске отображаются в диалоговом окне следующего вида(рис. 1.4).

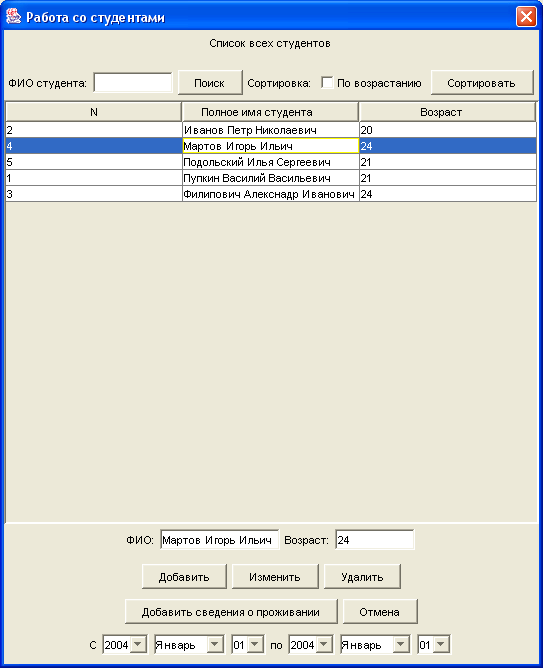


Рисунок 1.3 – Диалог для работы со списком студентов

Для завершения работы приложения клиента предназначен пункт меню «Выход». При завершении работы клиента посылается запрос на сервер, где закрывается выделенный для данного клиента поток и сокет, а также соединение, созданное для клиента для работы с базой данных.

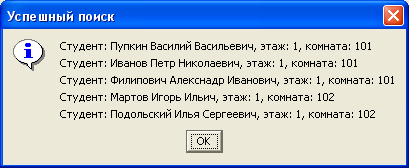


Рисунок 1.4 – Поиск сведений о проживании студентов

Для изменения настроек соединения с сервером используется диалог настроек (рис. 1.5). Для вступления изменений в силу требуется перезапуск приложения.

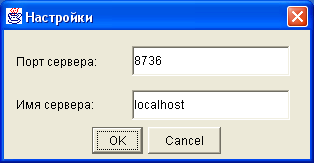


Рисунок 1.5 – Диалог настроек

Применение архитектуры «клиент-сервер» [8] предоставляет пользователям возможность с различных компьютеров обращаться к серверу за необходимой информацией. Сервер предоставляет возможность для клиента для работы с базой данных. Он выступает в качестве посредника между клиентом и базой данных. Он принимает запросы от клиента, их обрабатывает и направляет в базу данных. Также он перенаправляет запросы назад клиенту.

Клиент имеет возможность работы с информацией, хранимой в базе данных. Такая информация представлена в виде таблиц базы данных. Клиент может просматривать, редактировать, осуществлять поиск и т.д. необходимой ему информации.

Применение базы данных в качестве хранилища информации позволяет оптимально и эффективно хранить информацию, ее структурировать. Реализованный многопоточный сервер позволяет производить распараллеливание работы сразу нескольких клиентов в разных потоках.

**2** СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Изучив теоретические аспекты разрабатываемой системы и выработав список требований необходимых для разработки системы, разбиваем систему на функциональные блоки(модули). Это необходимо для обеспечения гибкой архитектуры. Такой подход позволяет изменять или заменять модули без изменения всей системы в целом.

В разрабатываемом веб-приложении можно выделить следующие блоки:

1. Блок авторизации.
2. Блок базы данных.
3. Блок пользовательского API.
4. Блок работы с БД.
5. Блок получения данных.
6. Блок работы с данными.
7. Блок обработки данных.
8. Блок «Ядро».

На каждый логически выделенный блок программы возлагаются определенные задачи. Кроме того, каждый блок программы так или иначе связан с некоторыми остальными блоками, чтобы обеспечить работоспособность всего приложения в целом. Связь, как правило, реализуется посредством обмена данными между блоками.

Согласно выбранной архитектуре – SPA (Single Page Application), основная часть логики приложения будет выполнятся на стороне клиента. Серверная часть лишь предоставляет доступ к ресурсам клиентской части приложения (html страница, JS файлы, таблицы стилей) и предоставляет API для работы базовыми операциями хранения данных приложения. Также на стороне сервера размещается блок авторизации. Блок авторизации используется для обеспечения авторизации пользователей и ограничения доступа к функциям приложения неавторизованных пользователей.

Приложение использует доступ к базе данных, соответственно выделен блок работы с базой данных. Таким образом, обобщённая структура приложения представлена на чертеже ГУИР.400201.012 C1.

*Блок пользовательского API* является основным блоком серверной части приложения. Данный блок предоставляет конечные точки для HTTP запросов. Также данный блок предоставляет 2 web-страницы: страницу авторизации и собственно страницу пользовательского интерфейса. Также данный блок поддерживает API методы, с помощью которых клиентская часть системы сможет загружать и сохранять данные. Данному блоку в Spring MVC соответствуют контроллеры, которые являются центральными компонентами в архитектуре MVC. Контроллер получает ввод пользователя, обрабатывает его и посылает обратно результат обработки, например, в виде представлений. Также к каждому данному блоку относятся наборы моделей и соответствующих представлений. Можно выделить основные команды, которые поддерживает данный блок:

1. получить страницу (включая html, JavaScript, CSS).
2. зарегистрировать пользователя.
3. авторизовать пользователя.
4. получить список студентов.
5. создать/получить/сохранить/удалить учетную запись студента.
6. получить/сохранить дисциплинарные взыскания студента.
7. получить список заселения комнат общежития.
8. получить список оценок комнат за текущий период.

*Блок авторизации* используется для ограничения доступа к функциям приложения неавторизованных пользователей. При первом доступе к приложению пользователю необходимо пройти регистрацию, где необходимо ввести логин и пароль. После этого данный пользователь сохраняется в системе и может использовать функции приложения.

Стоит отметить, что для регистрации и авторизации в приложении выделена отдельная страница, на которой пользователь должен ввести логин и пароль.

*Блок работы с базой данных* позволяет обращаться к таблицам в базе данных и выполнять с ними различные операции: чтения и поиск, вставка и удаление объектов.

*База данных* является центральным хранилищем данных в системе. В качестве системы управления базами данных (СУБД) было решено использовать MySQL. MySQL предназначена для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия, что является одним из факторов при выборе СУБД для использования в приложении.

*Блок «Ядро»* является основной частью клиентской части системы. Данный модуль отвечает за загрузку и совместную работу клиентских модулей. Каждый модуль связан с ядром, через которое происходит передача данных между блоками приложения. Также в данном блоке реализованы операции связи с сервером – здесь формируются HTTP запросы, результаты которых передаются соответствующему модулю приложения. Разработанная архитектура модуля «ядро» позволяет расширять функциональность системы путём добавления новых функциональных модулей.

*Блок получения данных* представляет собой модуль, осуществляющий транспортировку данных, полученных от авторизованного пользователя к блоку работы с введенными данными.

ЗАКЛЮЧЕНИЕ

На преддипломной практике передо мной была поставлена задача проектирования, разработки и тестирования программного средства учета и контроля студентов в общежитии. Целью программного средства является сокращение документооборота за счет построения системы сбора данных для статистики и аналитики актуальной информации о проживании студентов в общежитии.

За преддипломную практику было спроектировано следующее:

1. Проектирование блока базы данных.
2. Проектирование функциональных блоков программного средства.

Для разработки приложения выбран язык программирования Java, в качестве СУБД – MySQL.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Веб-сервис [Электронный ресурс]. – 2017. – Режим доступа : https://ru.wikipedia.org/wiki/Веб-служба.
2. Протокол HTTP [Электронный ресурс]. – 2017. – Режим доступа : https://ru.wikipedia.org/wiki/HTTP.
3. REST [Электронный ресурс]. – 2017. – Режим доступа : https://ru.wikipedia.org/wiki/REST.
4. Язык программирования Java. [Электронный ресурс]. – 2017. – Режим доступа : https://ru.wikipedia.org/wiki/Java.
5. Spring [Электронный ресурс]. – 2017. – Режим доступа : https://spring.io.
6. СУБД [Электронный ресурс]. – 2017. – Режим доступа : http://citforum.ru/database/articles/criteria/.
7. MySQL [Электронный ресурс]. – 2017. – Режим доступа : https://www.mysql.com.
8. Архитектура клиент-сервер [Электронный ресурс]. – 2017. – Режим доступа : https://ru.wikipedia.org/wiki/Клиент\_—\_сервер.

# ПРИЛОЖЕНИЕ А

(обязательное)

Вводный плакат. Плакат.

# ПРИЛОЖЕНИЕ Б

(справочное)

Программное средство учета и контроля студентов в общежитии. Схема структурная.