

Пермский филиал федерального государственного автономного  
образовательного учреждения высшего образования  
“Национальный исследовательский университет  
“Высшая школа экономики”

**Факультет профессиональной переподготовки**

## **Разработка системы «Сравнение компьютеров»**

---

Проектная работа по направлению подготовки  
«Объектно-ориентированное программирование»

Слушателя  
Группы

Калашникова А. Д.  
ООПД-17

Руководитель проектной работы

---

Старший преподаватель кафедры  
информационных технологий в бизнесе,  
Лебедев В. В.

Пермь, 2018

## **АННОТАЦИЯ**

Автор — Калашников Андрей Дмитриевич

Тема работы - «Разработка системы «Сравнение компьютеров»

Год издания — 2018

Издательство — Факультет профессиональной переподготовки НИУ ВШЭ —  
Пермь

Количество глав — 3

В работе рассмотрены вопросы, связанные с анализом, проектированием и созданием информационной системы подбора компьютерных комплектующих. Рассмотрены аналоги, их преимущества и недостатки.

Работа содержит 44 страницы основного текста, состоит из трех глав и трех приложений.

## Оглавление

Введение .....	6
Глава 1. Анализ предметной области .....	7
1.1. Оценка производительности компьютера .....	8
1.2. Обзор существующих решений .....	9
1.2.1. Программы оценки производительности конфигурации компьютера .....	9
1.2.2. Сервисы сравнения конфигураций компьютеров .....	11
1.3. Формирование требований к разрабатываемой системе.....	13
Глава 2. Проектирование программного решения .....	15
2.1. Диаграмма прецедентов.....	15
2.1.1. Прецедент «Вход в систему».....	16
2.1.2. Прецедент «Конструирование конфигурации» .....	17
2.1.3. Прецедент «Выбор центрального процессора» .....	17
2.1.4. Прецедент «Выбор оперативной памяти».....	18
2.1.5. Прецедент «Выбор платы видеоускорителя» .....	18
2.1.6. Прецедент «Выбор системы хранения данных».....	19
2.1.7. Прецедент «Обновление конфигурации».....	19
2.1.8. Прецедент «Поиск единицы оборудования базе данных» .....	20
2.1.9. Прецедент «Сравнение конфигураций» .....	20
2.1.10. Прецедент «Удаление конфигурации» .....	21
2.2. Диаграммы активностей .....	21
2.2.1 Вход в систему .....	21
2.2.2. Конструирование конфигурации .....	22
2.2.3. Обновление конфигурации .....	23
2.2.4. Поиск единицы оборудования в базе.....	24
2.2.5 Сравнение конфигураций .....	25
2.2.6. Удаление конфигурации .....	26
2.3. Статическая структура системы .....	28
2.3.1. Вход в систему .....	29
2.3.2. Конструирование конфигурации .....	29
2.3.3. Обновление конфигурации .....	29
2.3.4. Поиск единицы оборудования в базе данных.....	29

2.3.5. Сравнение конфигураций .....	30
2.3.6. Удаление конфигурации .....	30
2.4. Связи таблиц базы данных .....	30
2.5. Диаграммы последовательностей.....	32
2.5.1. Вход в систему .....	32
2.5.2. Конструирование конфигурации .....	32
2.5.3. Обновление конфигурации.....	33
2.5.4. Поиск единицы оборудования в базе данных.....	34
2.5.5. Сравнение конфигураций .....	34
2.5.6. Удаление конфигурации .....	35
Глава 3. Реализация информационной системы сравнения компьютеров .....	36
3.1. Технические средства реализации системы .....	36
3.2. Внешний вид разработанной системы .....	37
3.2.1. Вход в систему .....	37
3.2.2. Создание конфигурации.....	38
3.2.3. Сравнение конфигураций .....	39
3.2.4. Обновление конфигурации.....	39
3.2.5. Поиск единицы оборудования в базе данных.....	39
3.3. Тестирование информационной системы.....	41
Заключение .....	44
Библиографический список .....	45
Приложение А. Диаграммы классов.....	46
Приложение Б. Исходный код модульных тестов.....	55
Б.1. Модуль test_form.py .....	55
Б.2. Модуль test_models.py .....	56
Б.3. Модуль test_view.py .....	59
Приложение В. Исходный код программы .....	64
В.1. Модуль view.py .....	64
В.2. Модуль vergleich\urls.py .....	66
В.3. Модуль tables.py.....	67
В.4. Модуль models.py .....	68
В.5. Модуль forms.py.....	70

В.6. Модуль filters.py.....	70
В.7. Файл base.html.....	71
В.8. Файл compare_view.html .....	74
В.9. Файл computerconf_confirm_delete.html .....	76
В.10. Файл create_view.html.....	77
В.11. Файл index.html .....	78
В.12. Файл table_view.html .....	79
В.13. Модуль urls.py .....	81

## Введение

Современное общество невозможно представить без компьютеров. Будь то обычная сим-карта или дата-центр в несколько десятков гектар площадью, смартфоны, настольные решения, планшеты, игровые автоматы, сложные системы управления технологическими линиями — всё это представляет собой компьютер в том или ином виде. Компьютеризация, несомненно, затронула все сферы жизнедеятельности человечества. Вычислительные машины являются мощным инструментом, который упрощает нашу жизнь. Компьютеру не нужен отдых, а вышедшие из строя детали легко заменить. В последние годы складывается тенденция к развитию слабой форме искусственного интеллекта — когда компьютер начинает делать выводы по решаемой задаче самостоятельно, что уже применяется в области медицины и астрономии.

Среди массового потребителя очень популярно решение в виде стационарного домашнего компьютера, который позволяет выполнять игровые и мультимедийные функции. Чаще всего представляет собой совокупность нескольких компонентов: системный блок, монитор, манипуляторы ввода (клавиатура, мышь), колонки. Подобная конфигурация позволяет легко заменить любой из компонентов самостоятельно, не обладая специфичными знаниями или инструментом. Компоненты системного блока так же представляют из себя отдельные аппаратные решения, которые можно заменить. Если выбор периферийных устройств не представляет сложную задачу, то выбор компонентов системного блока — задача требующего особых знаний.

Цель данной работы: разработать информационную систему сравнения компьютеров. Основная задача, которая будет решаться: предоставить техническому специалисту средство, которое позволит сравнивать конфигурации компьютеров на основании объективной оценки.

## Глава 1. Анализ предметной области

Слово «компьютер» является англицизмом. В своей изначальной форме «computer» является производным от слова «compute», что означает «вычислять». Таким образом устройство, что позволяет производить вычисления является своеобразным компьютером. К первым устройствам подобного типа чаще всего относят счёты, которые появились примерно за три тысячи лет до нашей эры.

В конце второй мировой войны произошёл сильный скачок в развитии компьютеров в связи с необходимостью больших расчётов в рамках разработки ядерного вооружения [1]. В данный период появляются компьютеры первого поколения. Это были огромные вычислительные комплексы, основанные на электронных лампах. Второе поколение компьютеров относят к промежутку 1955 — 1965 г., с момента начала использования транзисторов. Третье поколение компьютеров (1965 — 1980 г.) появилось благодаря внедрению технологии интегральных схем. В данный период было создано легендарное семейство компьютеров IBM/360. Данное семейство стало популярным благодаря разделению архитектуры и реализации. Архитектура данного семейства, в свою очередь, стала промышленным стандартом и используется по сей день. Четвёртое поколение компьютеров (с 1980 г. по наши дни) связывают с появлением больших интегральных схем. В данный период компьютер начинает быть персонализированным устройством. Именно четвёртое поколение компьютеров и будет рассмотрено в рамках данной работы.

Компьютеры любого предшествующего поколения требовали большого штата специально обученных специалистов. При чём не только для возможности технического обслуживания, но и для ввода информации. На сегодняшний день собрать и настроить персональный компьютер может любой желающий. Чаще всего потребитель предпочитает переложить ответственность за качество конечного устройства на «плечи» продавца. Подобное разграничение ответственности даже привело к созданию организаций, которые занимаются подобной работой профессионально. Например, Dell.

Сегодня сердцем персональной компьютерной системы является системный блок. Эволюция электронных компонентов системного блока привела к тому, что на данный момент можно выделить следующие классы устройств [2]:

- Центральный процессор

- Материнская плата
- Оперативная память
- Плата графического ускорителя (видеокарта)
- Звуковая карта
- Сетевая карта
- Накопитель постоянной памяти (жёсткий или твердотельный диск)
- Привод оптических дисков
- Устройство считывания флеш-карт

Почти все компоненты системного блока влияют на производительность компьютера, но можно выделить основные узлы, которые принимают на себя основную вычислительную нагрузку: центральный процессор, оперативная память, плата графического ускорителя, жёсткий диск. Если обобщить выбранные компоненты, то это память и чипы, производящие все основные вычисления.

### **1.1. Оценка производительности компьютера**

Для объективной оценки производительности компьютера было создано специальное программное обеспечение — бенчмарки. Как правило бенчмарки представляют собой систему тестов производительности компьютера в повседневных задачах:

- Сжатие файлов архиватором
- Обработка мультимедиа контента
- Обработка объёмных сцен 3D-графики

Бывают и более специфичные тесты производительности:

- Тесты, проверяющие скорость записи и чтения в оперативной и постоянной памяти
- Тесты на скорость обработки математических вычислений

В ходе проведения теста программа формирует количественную оценку, которая обычно выражается в условных единицах. На усмотрение разработчика программы данная оценка может заноситься в базу данных, расположенной на сервере разработчика. Впоследствии программа может предоставить возможность сравнения текущей компьютерной конфигурации с конфигурациями из данной базы данных.



Существует так же количественная оценка вычислительной мощности компьютера — количество операций с числами с плавающей точкой в секунду (флопс; FLOPS англ.). Данная оценка, как правило, важна для сложных математических операций и операций моделирования.

## **1.2. Обзор существующих решений**

Сравнить конфигурации компьютеров возможно благодаря не только бенчмаркам, которые уже включает в функционал возможность сравнения текущей конфигурации с другими, но и с помощью открытых баз данных, составленных энтузиастами, или сервисом сравнения конфигураций, предоставляемым интернет-магазином.

### **1.2.1. Программы оценки производительности конфигурации компьютера**

Вот небольшой список бенчмарков, которые производят оценку производительности компьютера:

- 3DMark (см. рис. 1.1)
- PCMark
- AIDA64 (см. рис. 1.2)
- Fraps
- Jbenchmark

Программы для комплексного теста производительности компьютера, как правило, обеспечивают всесторонний тест, учитывающий все основные вычислительные узлы компьютера. По мере прохождения теста производительности на том или ином узле — программа выставляет оценку. После завершения теста пользователю предлагается оценить результат в виде итоговой оценки по всем проведённым тестам. По данной оценке можно сравнить производительность текущего компьютера с производительностью компьютеров других людей. Например, в сети интернет на сайтах компаний, производящий данный тест, или специализированных форумах. Если тест выдаёт количественную оценку производительности компьютера, выраженную в числах, то в разговорной речи употребляют термин «попугаи», поскольку не существует единицы измерения данной величины.

Подобная оценка является объективной, поскольку выполняется по заранее заложенному алгоритму и на результат влияет только быстродействие компонентов компьютера.

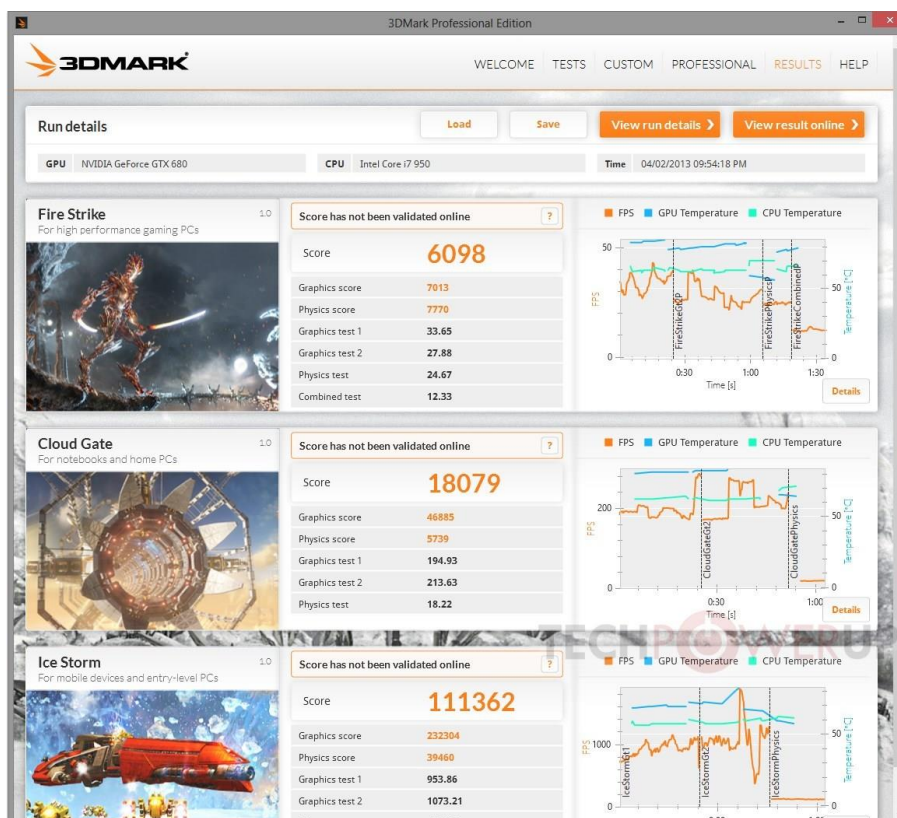


Рисунок 1.1. Основное окно программы "3DMark"

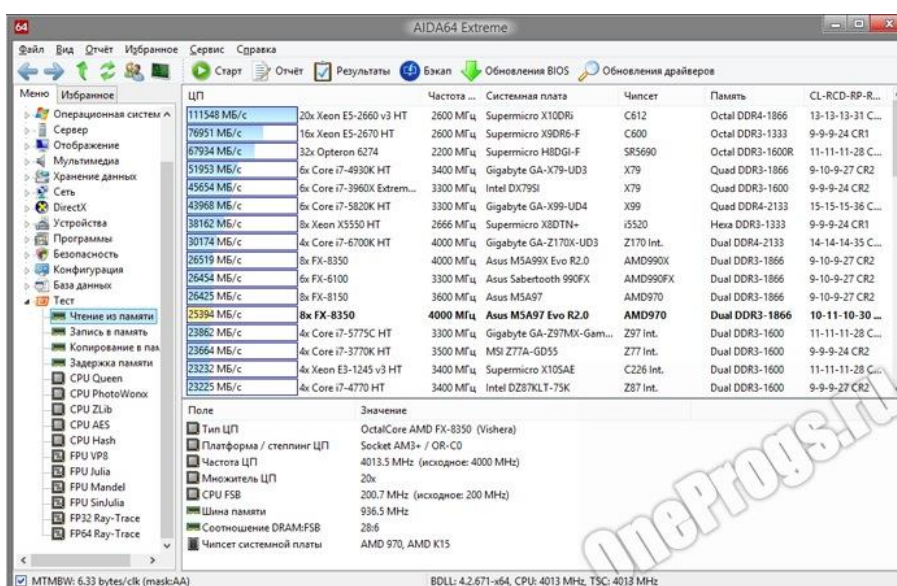


Рисунок 1.2. Окно программы «AIDA64» на вкладке тестирования оперативной памяти

С другой стороны, при использовании подобных программ возникает очень много факторов, которые влияют на конечную оценку:

- Изношенность компонентов системы
- Проблемы в системе охлаждения
- Используемая операционная система
- Запущенные программы на компьютере в момент теста

Как правило оценку в данных программах делают несколько раз. Даже один «проход» занимает продолжительное время, а совокупность подобных тестов увеличивает ожидание многократно.

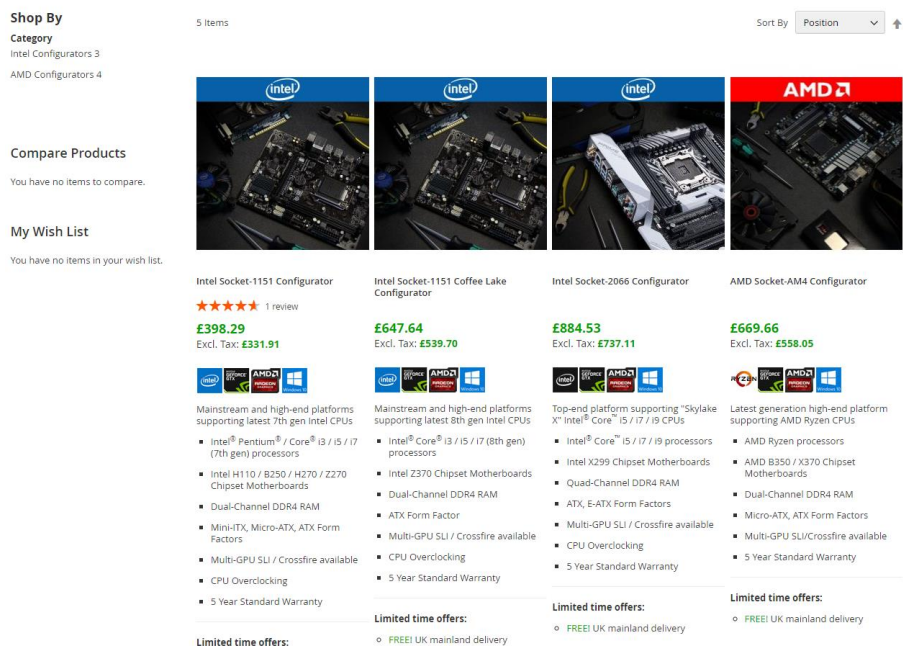
Использование одной и той же программы накладывает зависимость оценки от алгоритмов тестирования. Участники, которые хотят получить объективное сравнение, должны пользоваться одной и той же программой. Данное обстоятельство делает людей зависимыми от базы сравнения производительности компьютеров производителя данной программы. Если производитель данной программы по каким-либо причинам перестаёт поддерживать базу и данные в ней теряют актуальность — подобная оценка становится не показательной.

Качественное программное обеспечение (далее ПО) для тестирования компьютеров является платным и цены сравнительно высокие. Например, за одну лицензию профессионального издания программы PCMark было необходимо (на момент написания работы) заплатить порядка полутора тысячи долларов.

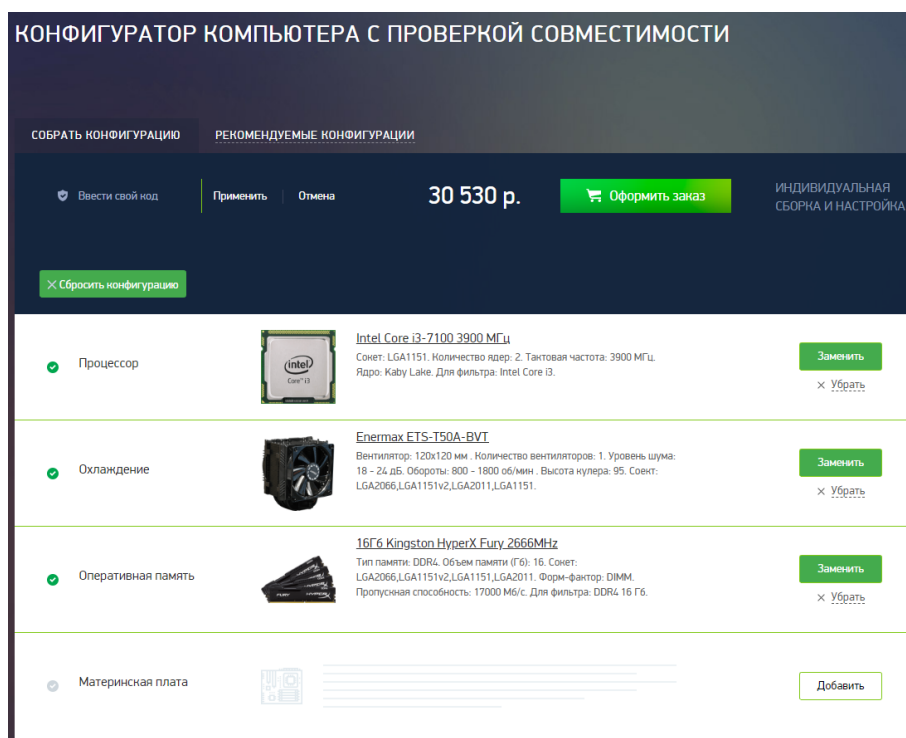
### **1.2.2. Сервисы сравнения конфигураций компьютеров**

Существует множество сервисов, которые позволяют конструировать электронную начинку будущего компьютера, а также сравнивать по объективной оценке различные конфигурации компьютера. Сервисы представлены как отечественными разработчиками, так и зарубежными:

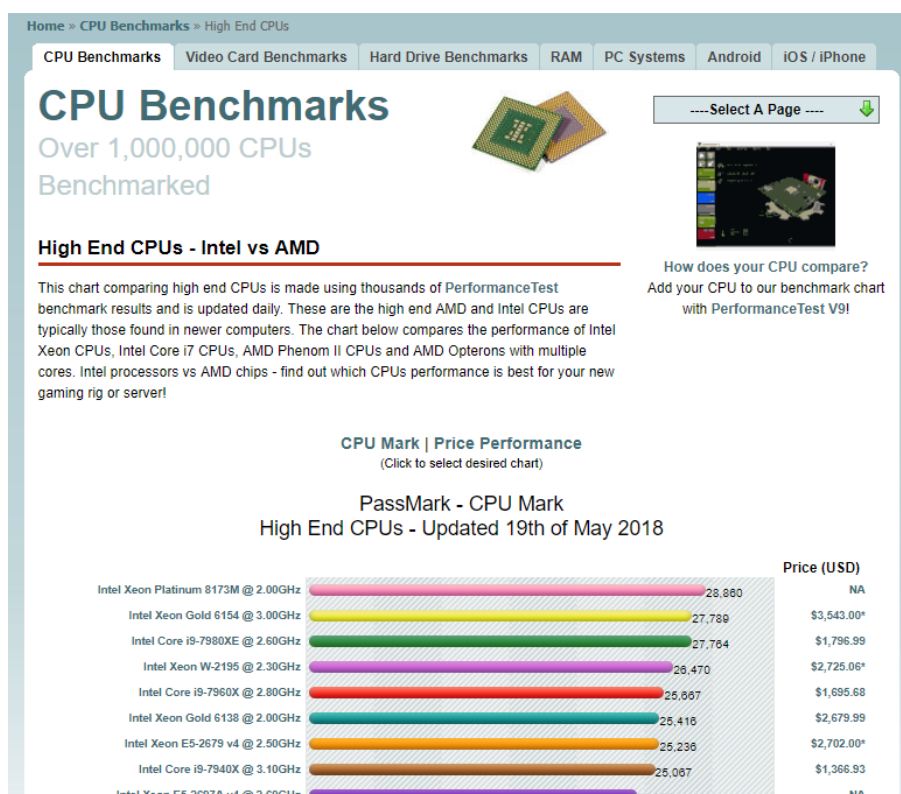
- <https://www.dinopc.com/online-pc-configurator/> (см. рис. 1.3)
- <https://edelws.ru/constructor/> (см. рис. 1.4)
- <http://www.ironbook.ru/constructor/>
- <https://www.cpubenchmark.net/> (см. рис. 1.5)



**Рисунок 1.3. Страница выбора конфигурации компьютера сайта «dinorcs.com»**



**Рисунок 1.4. Страница конструирования конфигурации компьютера сайта «edelws.ru»**



**Рисунок 1.5. Страница таблицы результатов бенчмарков центральных процессоров на сайте «scribenchmark.net»**

Подобные сервисы, как правило, предоставляют магазины, чтобы покупатель после подбора интересующего оборудования мог сразу его купить, что, несомненно, является плюсом. Основным недостатком подобных сервисов является отсутствие возможности оценить производительность получаемой конфигурации в сравнении с другими конфигурациями. Также при отсутствии интернет-соединения сервис будет недоступен.

### **1.3. Формирование требований к разрабатываемой системе**

На основании всего вышеизложенного функциональные требования к разрабатываемой системе будут следующими:

- Разрабатываемая система должна предоставлять возможность сравнивать получаемые конфигурации компьютеров на основании объективной оценки производительности
- Оценка производительности узлов компьютера должна производиться на основании числового значения, полученного с помощью специализированного ПО, либо из пополняемой актуальной общедоступной базы данных подобных оценок

- Разрабатываемая система должна иметь связь с существующей базой доступного оборудования на складе предприятия
- Доступ к системе должен осуществляться вне зависимости от программной платформы (операционной системы)

## **Глава 2. Проектирование программного решения**

Этап проектирования программного решения позволит сформулировать и структурировать основные решаемые задачи в ходе разработки, выделить основные группы пользователей системы, описать поведение системы в разных состояниях.

В качестве инструмента для проектирования будет использоваться унифицированный язык моделирования (англ. universal modeling language, UML). Язык представляет собой набор правил, которые позволяют создать диаграммы [3], отражающие суть решаемых задач и проблем, взаимодействие компонентов системы и т.д.

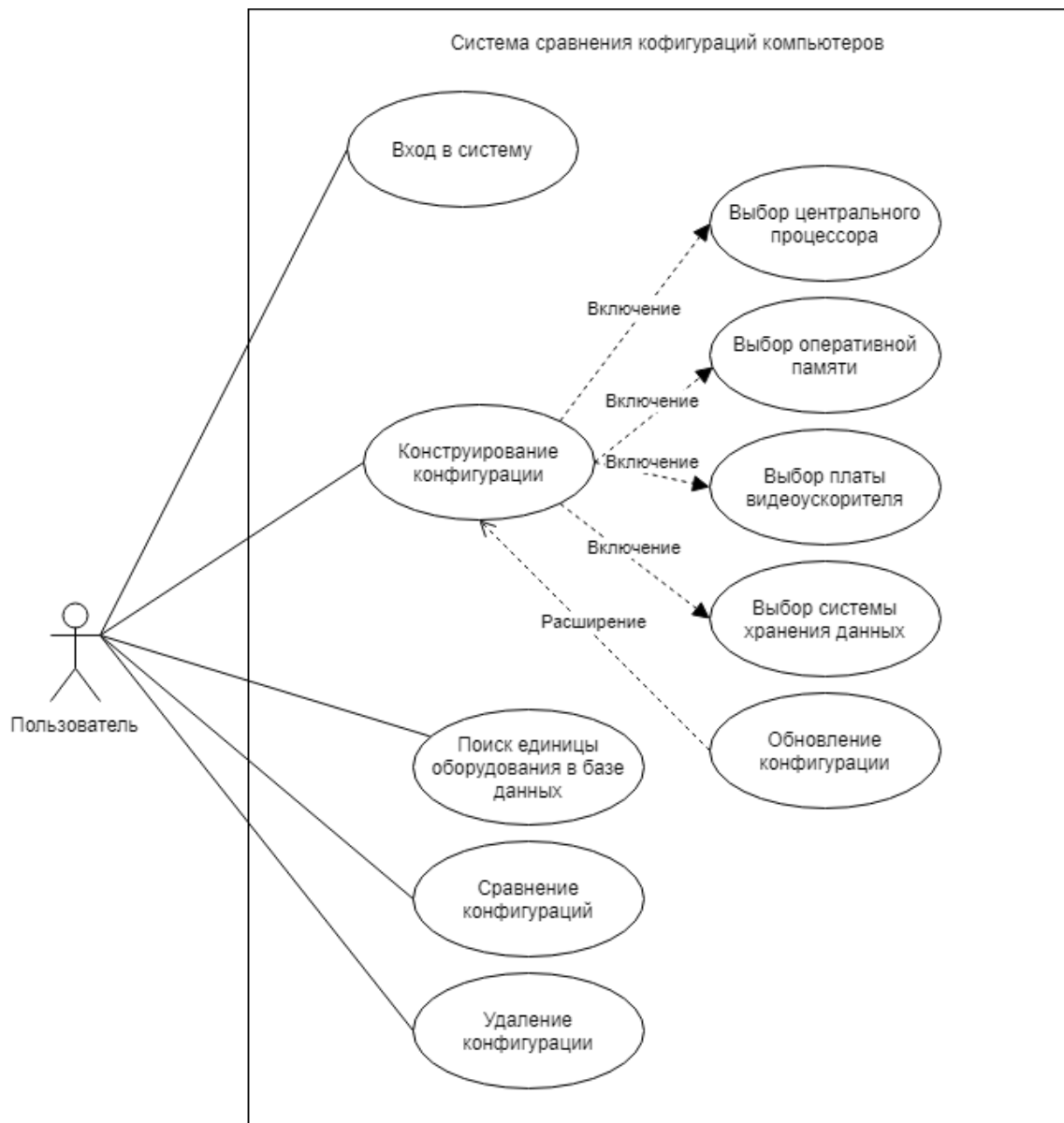
### **2.1. Диаграмма прецедентов**

Диаграмма прецедентов показывает основных пользователей будущего программного обеспечения и основные варианты использования. Для разрабатываемой системы диаграмма прецедентов приведена ниже (см. рис. 2.1).

Пользователем системы является технический специалист, который может войти в данную систему и производить различные манипуляции с конфигурациями:

- Создавать
- Сохранять
- Сравнивать одни конфигурации с другими
- Удалять конфигурации
- Обновлять сохранённые ранее конфигурации

Конструирование конфигурации, в свою очередь, представляет из себя последовательность операций выбора компонентов.



**Рисунок 2.1. Диаграмма прецедентов**

### **2.1.1. Прецедент «Вход в систему»**

Название: вход в систему.

Актор: пользователь.

Краткое описание: пользователь авторизуется в системе.

Триггер: открытие главного окна программы.



**Таблица 2.1. Вход в систему**

Действие актора	Отклик системы
Заходит в систему	Формирует страницу авторизации
Вводит имя пользователя и пароль (E1)	Предоставление доступа к основной странице программы

Альтернативные потоки:

E1: В случае некорректного ввода имени пользователя или пароля система перенаправляет пользователя на повторный ввод данных.

### **2.1.2. Прецедент «Конструирование конфигурации»**

Название: конструирование конфигурации.

Актор: пользователь.

Краткое описание: создание, изменение, просмотр и удаление конфигурации компьютера.

Триггер: пользователь выполнил вход в систему и перешёл на соответствующую вкладку программы

**Таблица 2.2. Конструирование конфигурации**

Действие актора	Действие актора
Актор последовательно производит выбор единиц оборудования для составления конфигурации.	Система фиксирует выбор пользователя.
Актор посылает сигнал на сохранение конфигурации.	Сохраняет созданную конфигурацию (E1, E2).

Подпотоки: отсутствуют.

Альтернативные потоки:

E1: Если пользователь не выбрал единицу оборудования из представленных позиций — возникает предупреждение о необходимости заполнить соответствующее поле.

E2: Если количество возможных сохранённых конфигураций превышает заданное — система выдаёт предупреждение о невозможности сохранить текущую собранную конфигурацию.

### **2.1.3. Прецедент «Выбор центрального процессора»**

Название: выбор центрального процессора.

Актор: пользователь.

Краткое описание: пользователю предоставляется возможность выбора модели центрального процессора.

Триггер: пользователь выполнил вход в систему и перешёл на соответствующую вкладку программы.

**Таблица 2.3. Выбор центрального процессора**

Действие актора	Действие актора
Актор посылает сигнал на предоставление возможности выбора модели центрального процессора.	Актор посылает сигнал на предоставление возможности выбора модели центрального процессора.

Подпотоки: отсутствуют.

Альтернативные потоки: отсутствуют.

#### **2.1.4. Прецедент «Выбор оперативной памяти».**

Название: выбор оперативной памяти.

Актор: пользователь.

Краткое описание: пользователю предоставляется возможность выбора модели оперативной памяти.

Триггер: пользователь выполнил вход в систему и перешёл на соответствующую вкладку программы.

**Таблица 2.4. Выбор оперативной памяти**

Действие актора	Отклик системы
Актор посылает сигнал на предоставление возможности выбора модели оперативной памяти.	Система делает запрос к базе данных и предоставляет пользователю выборку моделей оперативной памяти.

Подпотоки: отсутствуют.

Альтернативные потоки: отсутствуют.

#### **2.1.5. Прецедент «Выбор платы видеоускорителя»**

Название: выбор платы видеоускорителя.

Актор: пользователь.

Краткое описание: пользователю предоставляется возможность выбора модели платы видеоускорителя.

Триггер: пользователь выполнил вход в систему и перешёл на соответствующую вкладку программы.

**Таблица 2.5. Выбор оперативной памяти**

Действие актора	Отклик системы
Актор посылает сигнал на предоставление возможности выбора модели платы видеоускорителя.	Система делает запрос к базе данных и предоставляет пользователю выборку моделей платы видеоускорителя.

Подпоток: отсутствуют.

Альтернативные потоки: отсутствуют.

### **2.1.6. Прецедент «Выбор системы хранения данных»**

Название: выбор системы хранения данных.

Актор: пользователь.

Краткое описание: пользователю предоставляется возможность выбора системы хранения данных.

Триггер: пользователь выполнил вход в систему и перешёл на соответствующую вкладку программы.

**Таблица 2.6. Выбор оперативной памяти**

Действие актора	Отклик системы
Актор посылает сигнал на предоставление возможности выбора системы хранения данных.	Система делает запрос к базе данных и предоставляет пользователю выборку систем хранения данных.

Подпоток: отсутствуют.

Альтернативные потоки: отсутствуют.

### **2.1.7. Прецедент «Обновление конфигурации»**

Название: обновление конфигурации.

Актор: пользователь.

Краткое описание: пользователю предоставляется возможность изменить выбранную конфигурацию.

Триггер: пользователь выполнил вход в систему и перешёл на соответствующую вкладку программы.

**Таблица 2.7. Обновление конфигурации**

<b>Действие актора</b>	<b>Отклик системы</b>
Актор посылает сигнал на предоставление возможности обновления конфигурации.	Система формирует ответ в виде формы конструирования конфигурации с подставленными значениями из выбранной конфигурации.

Подпотоки: отсутствуют.

Альтернативные потоки: отсутствуют.

### **2.1.8. Прецедент «Поиск единицы оборудования базе данных»**

Название: поиск единицы оборудования базе данных.

Актор: пользователь.

Краткое описание: поиск единицы оборудования в текущей базе данных с заданными критериями.

Триггер: пользователь выполнил вход в систему и перешёл на соответствующую вкладку программы.

**Таблица 2.8. Поиск единицы оборудования**

<b>Действие актора</b>	<b>Отклик системы</b>
Актор вводит критерии поиска.	Система формирует запрос к базе данных с заданными критериями.
Актор посылает сигнал о начале поиска.	Система обрабатывает сигнал и предоставляет пользователю соответствующую выборку.

Подпотоки: отсутствуют.

Альтернативные потоки: отсутствуют.

### **2.1.9. Прецедент «Сравнение конфигураций»**

Название: сравнение конфигураций.

Актор: пользователь.

Краткое описание: система предоставляет возможность просмотра всех составленных конфигураций для анализа пользователем.

Триггер: пользователь выполнил вход в систему и перешёл на соответствующую вкладку программы.

**Таблица 2.9. Сравнение конфигураций**

Действие актора	Отклик системы
Актор посылает запрос на вывод информации о текущих составленных конфигурациях.	Система формирует подробную выборку о составленных конфигурациях и предоставляет сформированный ответ для пользователя.

Подпотоки: отсутствуют.

Альтернативные потоки: отсутствуют.

### **2.1.10. Прецедент «Удаление конфигурации»**

Название: удаление конфигурации.

Актор: пользователь.

Краткое описание: система предоставляет возможность удалить выбранную конфигурацию.

Триггер: пользователь выполнил вход в систему и перешёл на соответствующую вкладку программы.

**Таблица 2.10. Сравнение конфигураций**

Действие актора	Отклик системы
Актор посылает запрос на удаление выбранной конфигурации.	Система фиксирует выбранную конфигурацию и формирует запрос на подтверждение проведения операции.
Актор посылает сигнал о подтверждении(E1).	Система удаляет выбранную конфигурацию из общего списка.

Подпотоки: отсутствуют.

Альтернативные потоки:

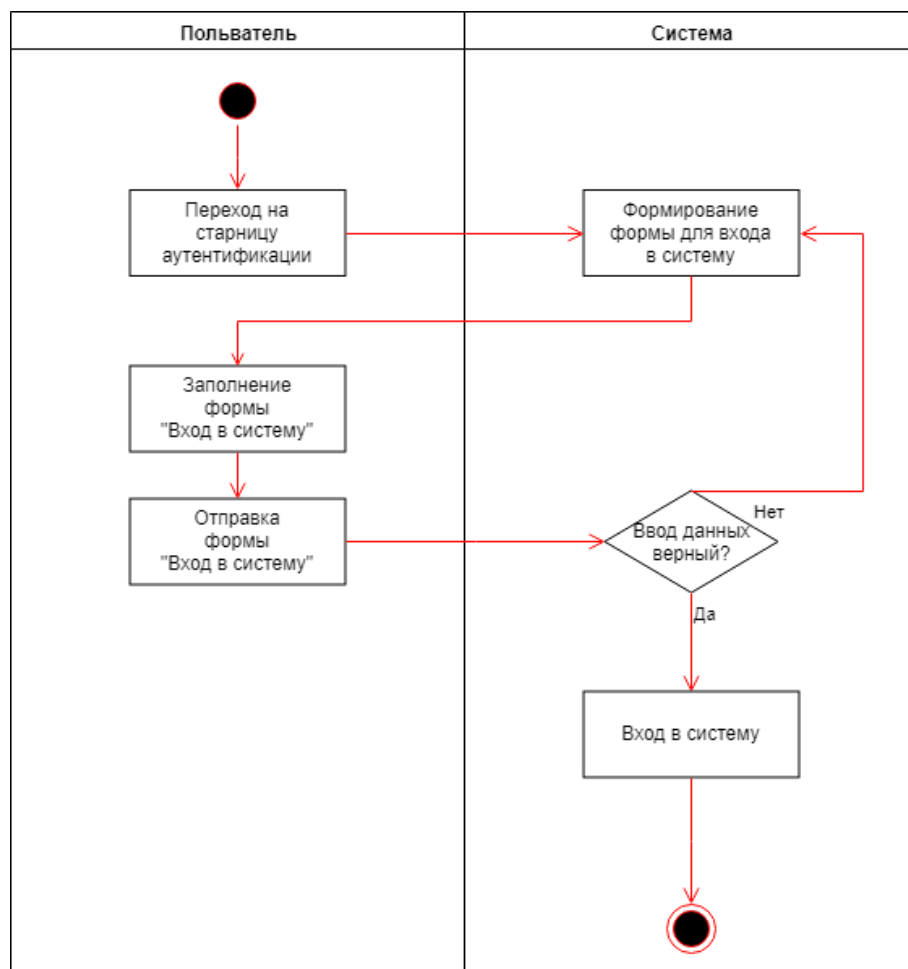
E1: Если актор изменил своё решение и посылает сигнал об отмене операции удаления — система перенаправляет актора на страницу сравнения конфигураций.

## **2.2. Диаграммы активностей**

Диаграммы активностей демонстрируют процесс взаимодействия пользователя и системы, а наглядное представление упрощает восприятие информации и систематизацию процессов, происходящих в системе.

### **2.2.1 Вход в систему**

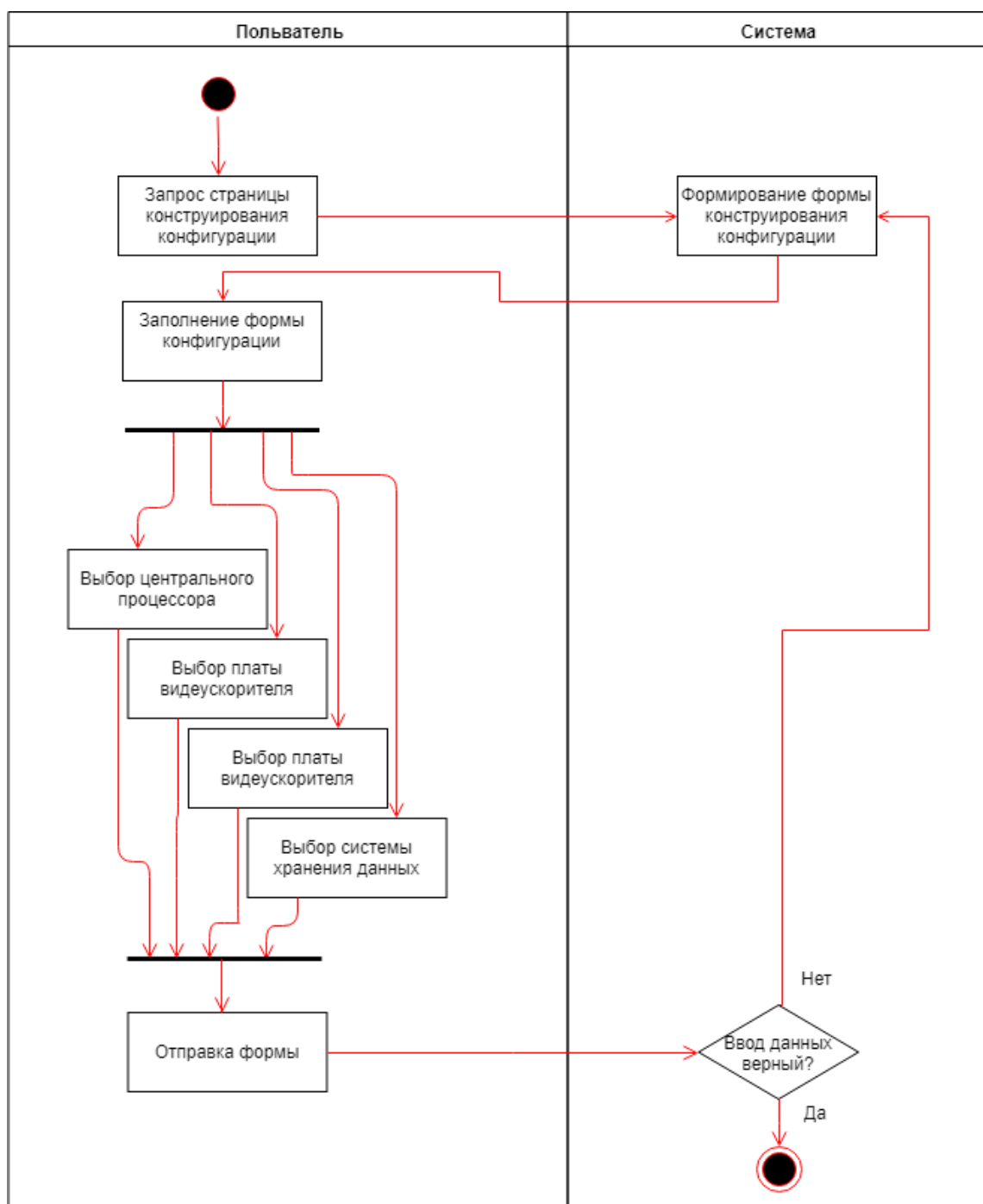
Диаграмма активностей «Вход в систему» отображает порядок действий пользователя системы и системы. Диаграмма представлена на рисунке ниже (см. рис. 2.2)



**Рисунок 2.2** Диаграмма «Вход в систему»

### 2.2.2. Конструирование конфигурации

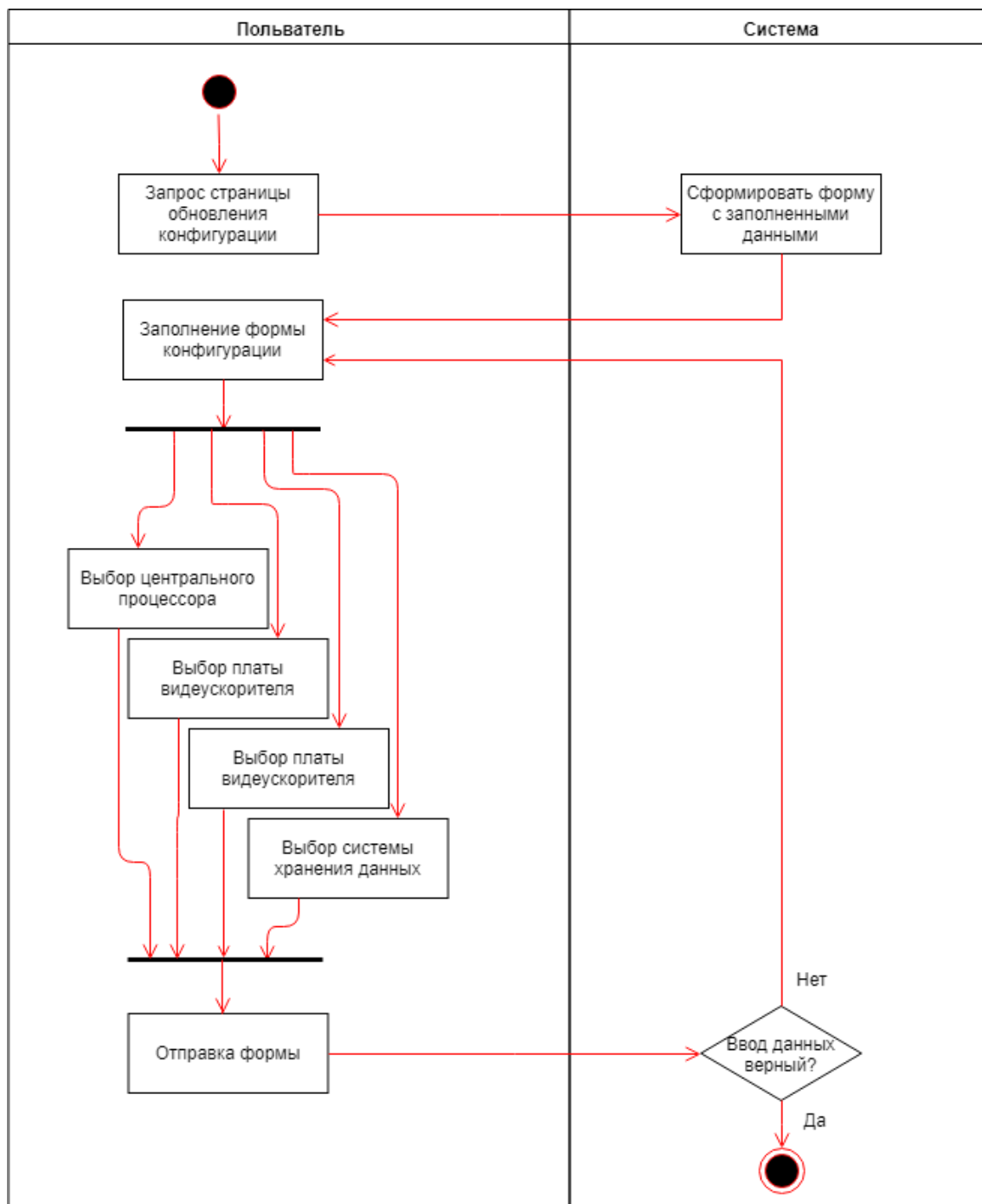
Диаграмма активностей «Конструирование конфигурации» отображает порядок действий пользователя системы и системы. Диаграмма представлена на рисунке ниже (см. рис. 2.3).



**Рисунок 2.3** Диаграмма «Конструирование конфигурации»

### 2.2.3. Обновление конфигурации

Диаграмма активностей «Обновление конфигурации» отображает порядок действий пользователя системы и системы. Диаграмма представлена на рисунке ниже (см. рис. 2.4).

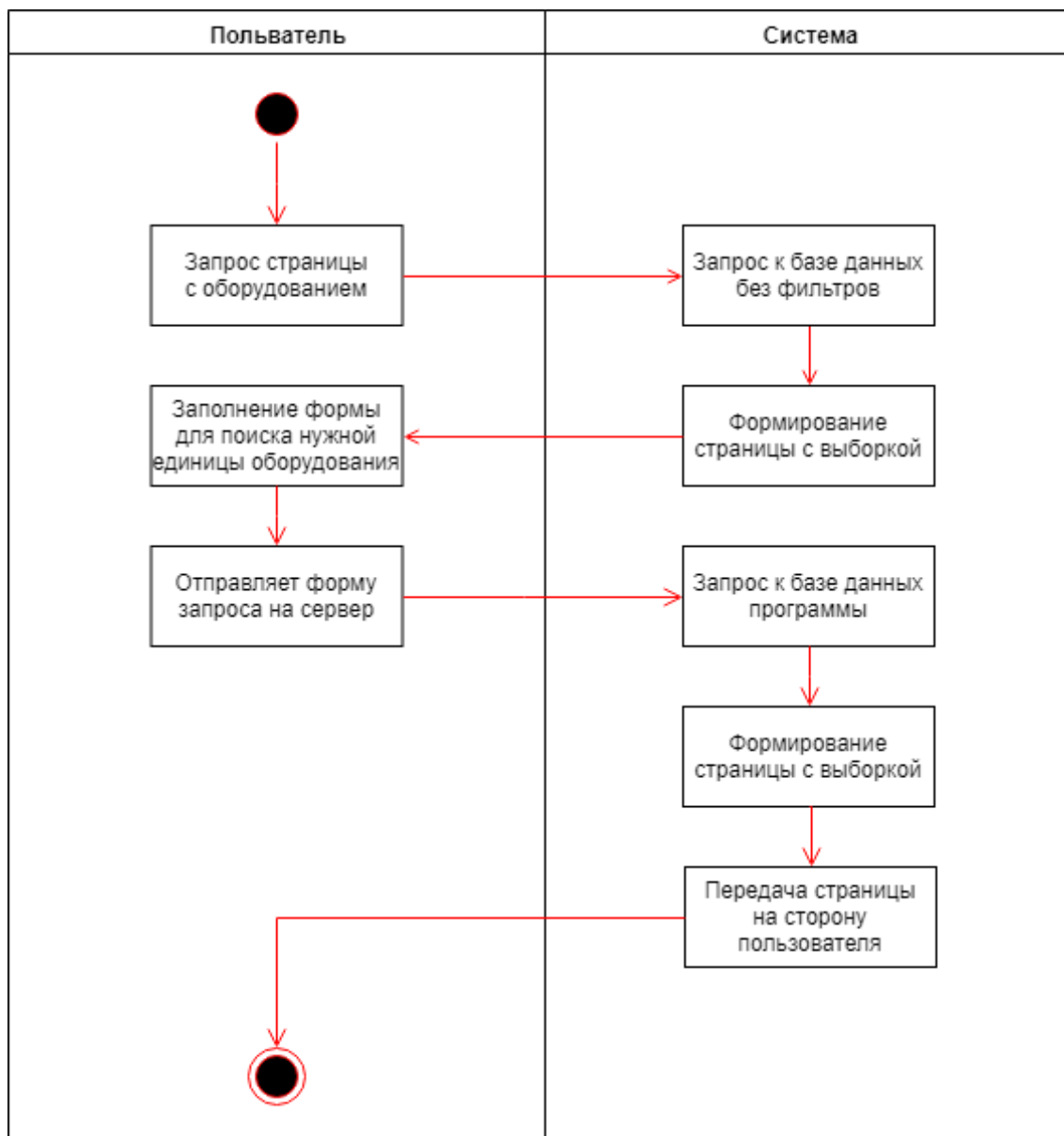


**Рисунок 2.4. Диаграмма «Обновление конфигурации»**

#### **2.2.4. Поиск единицы оборудования в базе**

Диаграмма активностей «Поиск единицы оборудования в базе» отображает порядок действий пользователя системы и системы. Диаграмма представлена на рисунке ниже (см. рис. 2.5).

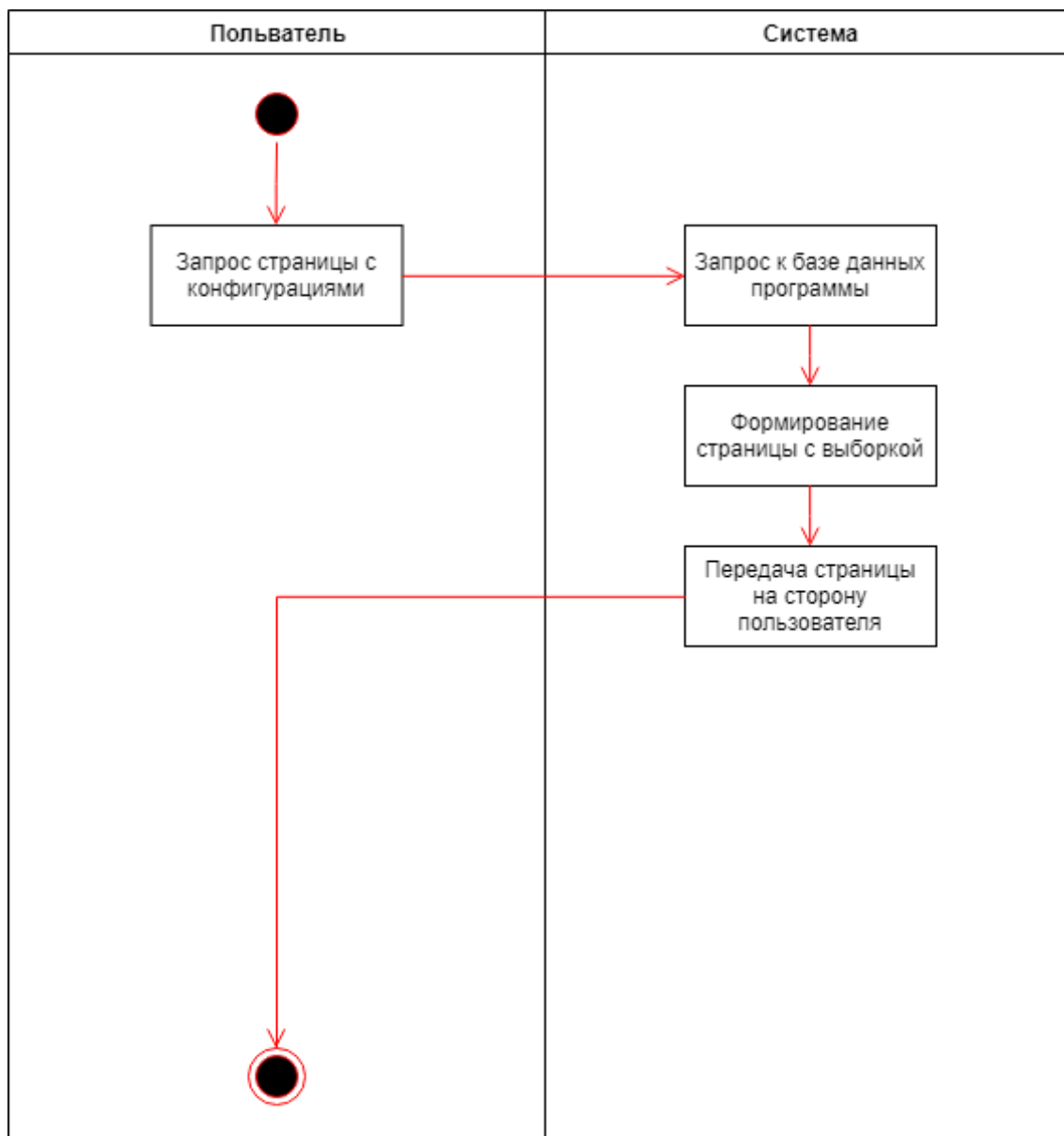




*Рисунок 2.5. Диаграмма «Поиск единицы оборудования в базе»*

### 2.2.5 Сравнение конфигураций

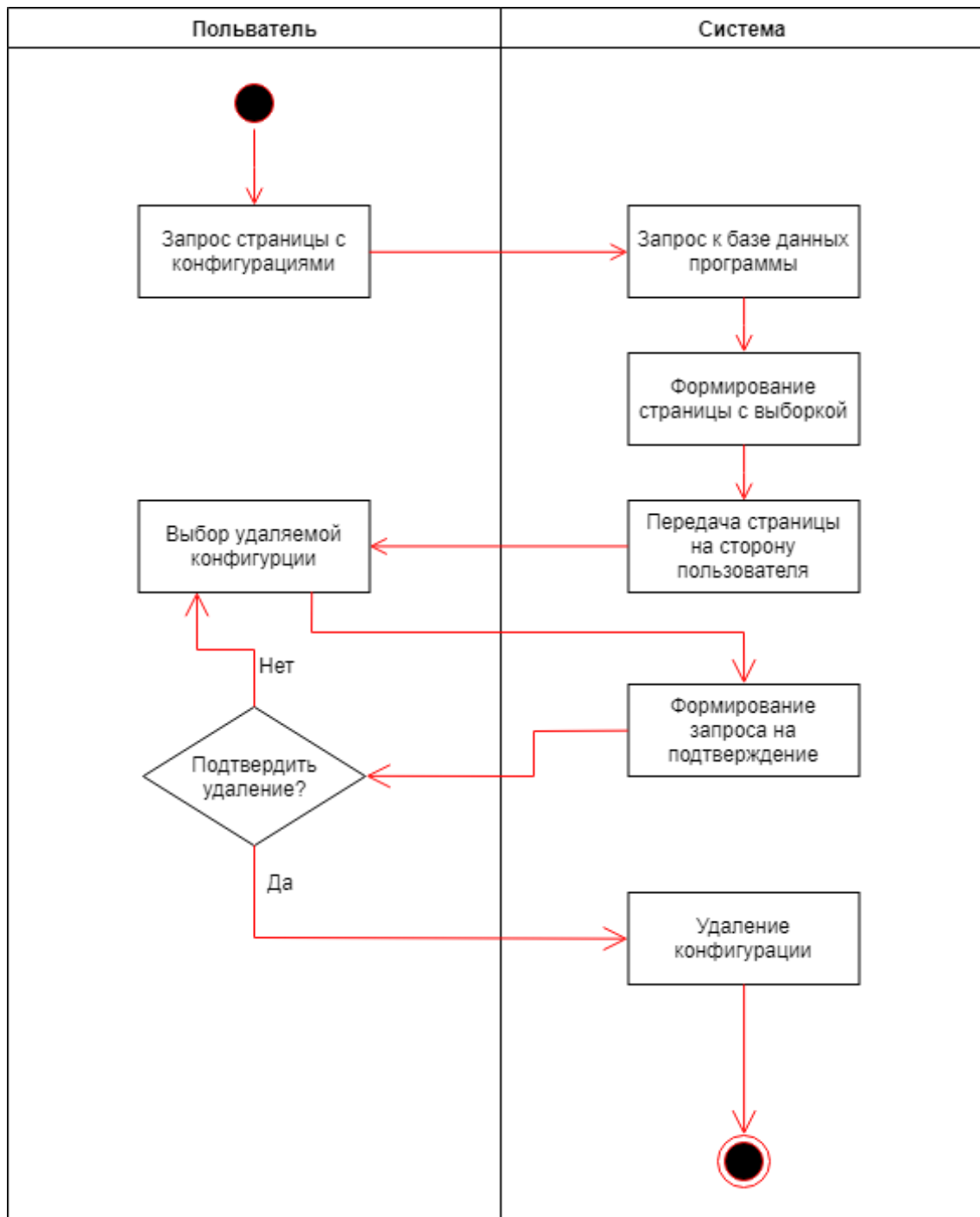
Диаграмма активностей «Сравнение конфигураций» отображает порядок действий пользователя системы и системы. Диаграмма представлена на рисунке ниже (см. рис. 2.6).



*Рисунок 2.6. Диаграмма «Сравнение конфигураций»*

### **2.2.6. Удаление конфигурации**

Диаграмма активностей «Удаление конфигурации» отображает порядок действий пользователя системы и системы. Диаграмма представлена на рисунке ниже (см. рис. 2.7).



**Рисунок 2.7. Диаграмма “Удаление конфигурации”**

## 2.3. Статическая структура системы

Разрабатываемая система будет представлять из себя веб-приложение. Приложения подобного типа содержат в себе две части [4]:

- Фронтенд (frontend англ. - передняя часть) — представляющий из себя веб-страницы.
- Бэкенд (backend англ. - задняя часть) — представляющий из себя совокупность программ, обрабатывающих всю основную логику приложения на стороне сервера.

Фронтенд может содержать в себе сложную логику, но она, как правило, относится к тому — каким образом информация будет отображаться в браузере пользователя. Фронтенд определяется стеком трёх технологий: HTML – гипертекстовый язык разметки страниц; CSS – набор каскадных стилей для определения стиля объектов на странице; JavaScript – интерпретируемый язык программирования, который предоставляет возможность написания сложной логики взаимодействия элементов на странице. В рамках данного проекта фронтенд определяет пользовательский интерфейс, который будет описан в соответствующем разделе.

Бэкенд составляет «скелет» проекта, который выполняется на стороне сервера и осуществляет всю основную работу по обработке данных: взаимосвязь с базой данных, взаимосвязь со сторонними модулями и программами, подготовка данных для передачи их в контекст страницы и т. д. Для того, чтобы разделить ответственность выполнения тех или иных операций широко применяется паттерн проектирования приложений «MVC».

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо [5].

В рамках данного проекта модель будет представлять логику, которая отвечает за взаимосвязь с базой данных. Описывая объект модели, фактически, мы описываем и сущность таблицы, в которую будут заноситься данные для приложения.

Объекты представления — это страницы с гипертекстовой разметкой со специальными управляющими символами, обрабатывая которые, сервер подставляет требуемую информацию.

Объекты контроллера — сущности, которые соединяют логику работы с объектами модели и представления. Данные объекты отвечают за обращение к объектам модели, чтобы те в свою очередь предоставили информацию из базы данных. Обрабатывают данную информацию в соответствии с требуемой логикой работы приложения и передают результат в контекст объектов представления.

В соответствии с принятым паттерном и диаграммой прецедентов составим схемы взаимодействия объектов в нотации диаграмм классов UML.

### **2.3.1. Вход в систему**

Связь сущностей представляет собой реализацию ограничение прав доступа пользователя на страницу. Основные классы контроллеров наследуют свойства базового класса аутентификации, который реализует в себе основную логику. Диаграмма классов представлена на рисунке в приложении (см. прил. А, рис.А.1).

### **2.3.2. Конструирование конфигурации**

Связь сущностей представляет собой реализацию возможности добавления конфигурации компьютера к сравнению. При этом главный объект модели связан с другими моделями и хранит в себе лишь ссылку на объекты других моделей. Диаграмма классов представлена на рисунке в приложении (см. прил. А, рис.А.2).

### **2.3.3. Обновление конфигурации**

Связь сущностей представляет собой реализацию возможности обновления конфигурации компьютера. При этом главный объект модели связан с другими моделями и хранит в себе лишь ссылку на объекты других моделей. Диаграмма классов представлена на рисунке в приложении (см. прил. А, рис.А.3).

### **2.3.4. Поиск единицы оборудования в базе данных**

В проекте используются четыре типа аппаратного обеспечения компьютера:

- Центральный процессор
- Оперативная память
- Жёсткий диск

- Видеоускоритель

Рационально разделить поиск единицы оборудования по каждой из категории и для каждой категории сформировать свою отдельную таблицу в базе данных, а следовательно, предусмотреть отдельную модель. Ранее было показано, что относительно каждой из моделей будет существовать связь с моделью сравнения конфигураций. В связи с этим и принятым соглашением об использовании паттерна MVC ниже будут представлены следующие диаграммы связей:

- Диаграмма классов «Поиск в БД проекта моделей центральных процессоров» (см. прил. А, рис.А.4).
- Диаграмма классов «Поиск в БД проекта моделей оперативной памяти» (см. прил. А, рис.А.5).
- Диаграмма классов «Поиск в БД проекта моделей жёстких дисков» (см. прил. А, рис.А.6).
- Диаграмма классов «Поиск в БД проекта моделей видеоускорителей» (см. прил. А, рис.А.7).

### **2.3.5. Сравнение конфигураций**

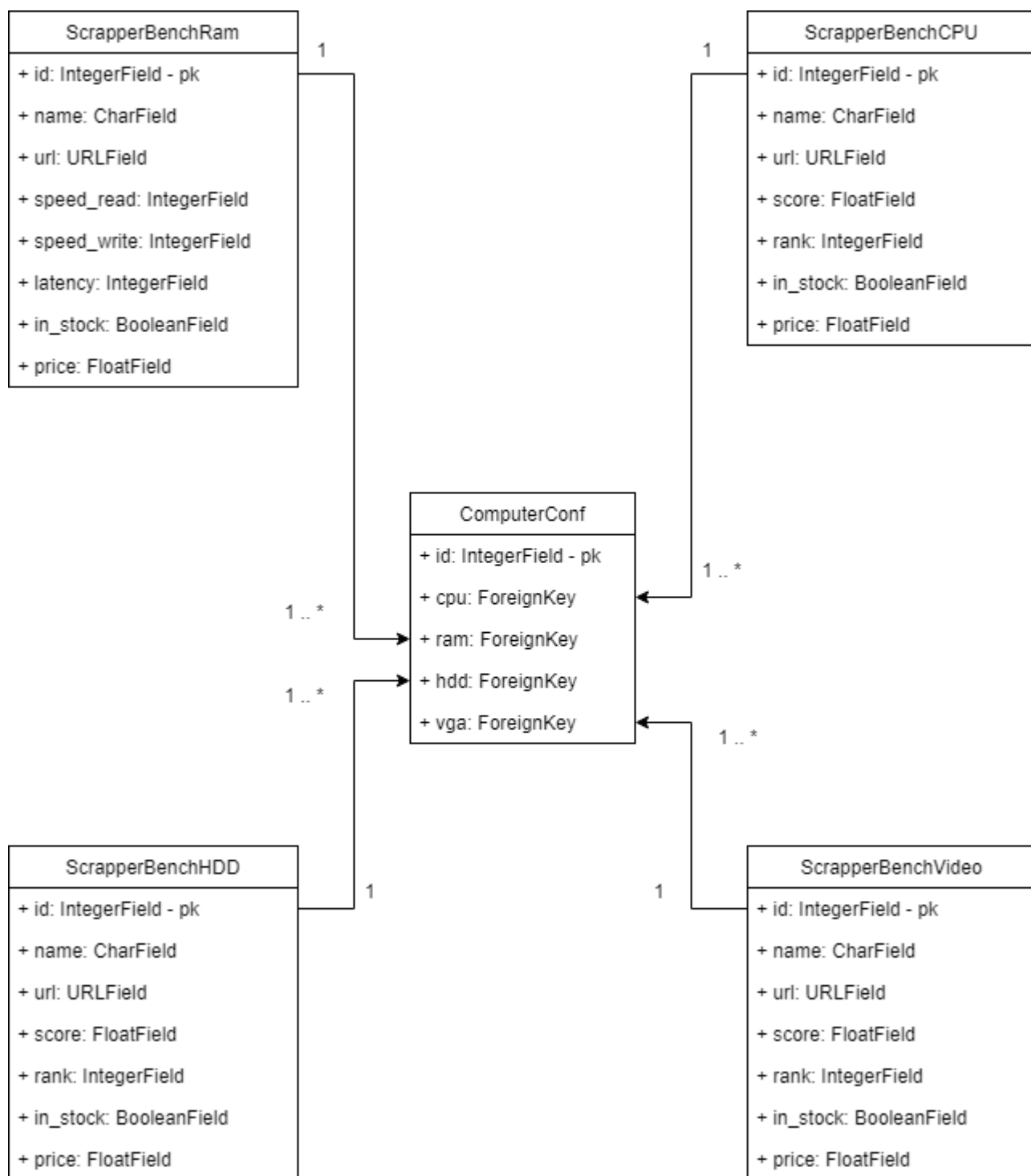
Связь сущностей представляет собой реализацию возможности сравнения ранее добавленных конфигураций компьютера. При этом главный объект модели связан с другими моделями и хранит в себе лишь ссылку на объекты других моделей. Диаграмма классов представлена на рисунке в приложении (см. прил. А, рис.А.8).

### **2.3.6. Удаление конфигурации**

Связь сущностей представляет собой реализацию возможности удаления ранее добавленных конфигураций компьютера. При этом главный объект модели связан с другими моделями и хранит в себе лишь ссылку на объекты других моделей. Диаграмма классов представлена на рисунке в приложении (см. прил. А, рис.А.9).

## **2.4. Связи таблиц базы данных**

Общая структура таблиц базы данных разрабатываемой системы представлена ниже (см. рис. 2.8). На представленной диаграмме связей отображена связь между таблицами, а также типы применяемых полей.



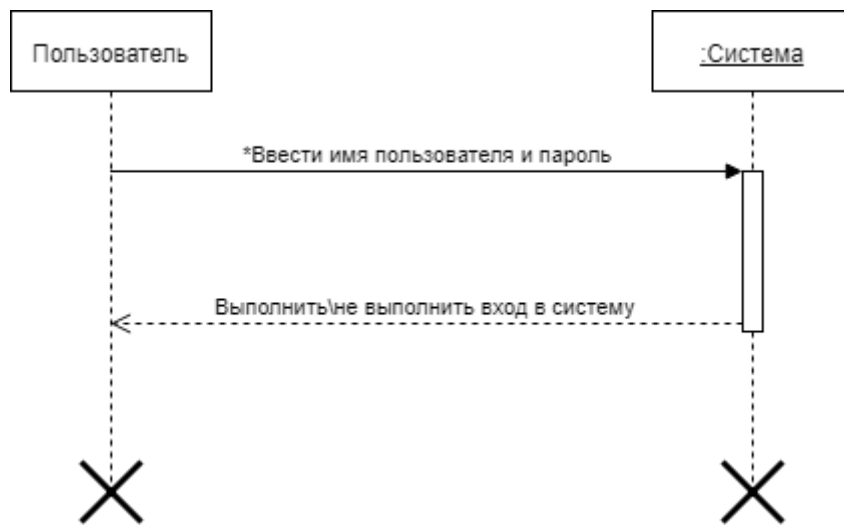
*Рисунок 2.8. Диаграмма связей таблиц в базе данных*

## 2.5. Диаграммы последовательностей

Диаграммы, на которых показано взаимодействие акторов в рамках какого-либо определённого прецедента называются диаграммами последовательностей. Ниже представлены диаграммы последовательностей взаимодействия пользователя и разрабатываемой системы.

### 2.5.1. Вход в систему

Обмен сообщениями при входе в систему представлен на рисунке 2.9.



*Рисунок 2.9. Вход в систему*

### 2.5.2 Конструирование конфигурации

Обмен сообщениями при конструировании конфигурации представлен на рисунке ниже (см. рис. 2.10).





*Рисунок 2.10. Конструирование конфигурации*

### 2.5.3. Обновление конфигурации

Обмен сообщениями при обновлении конфигурации представлен на рисунке 2.11.



*Рисунок 2.11. Обновление конфигурации*

#### 2.5.4. Поиск единицы оборудования в базе данных

Обмен сообщениями при поиске единицы оборудования в базе данных представлен на рисунке 2.12.

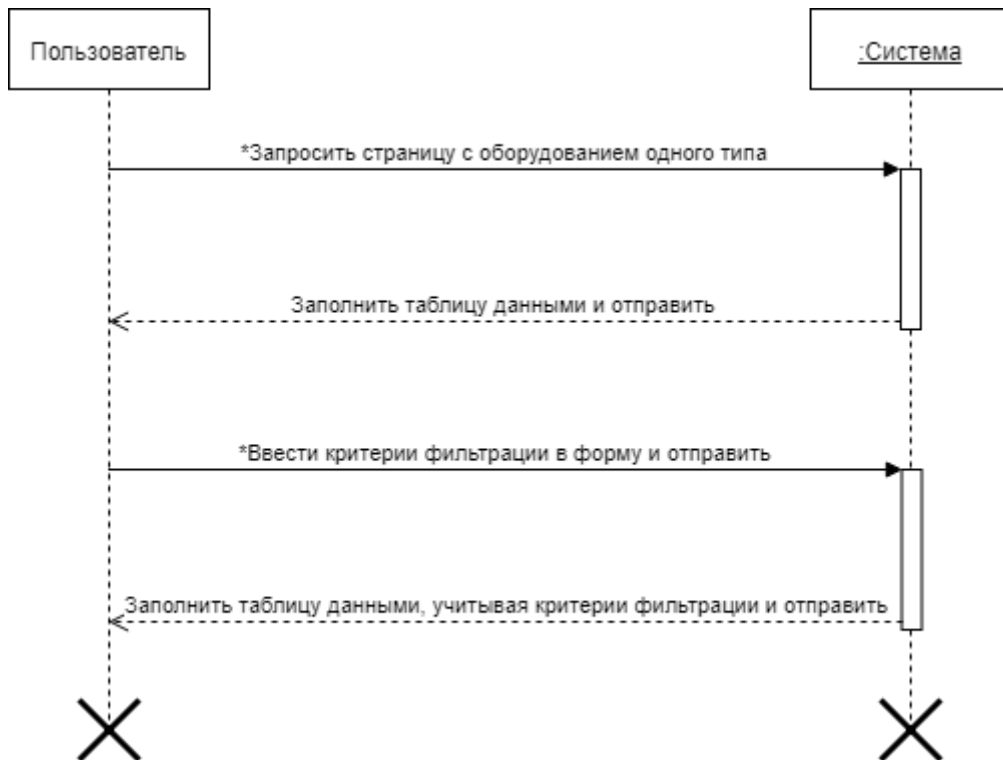


Рисунок 2.12. Поиск единицы оборудования в базе данных

#### 2.5.5. Сравнение конфигураций

Обмен сообщениями при сравнении конфигураций представлен на рисунке 2.13.

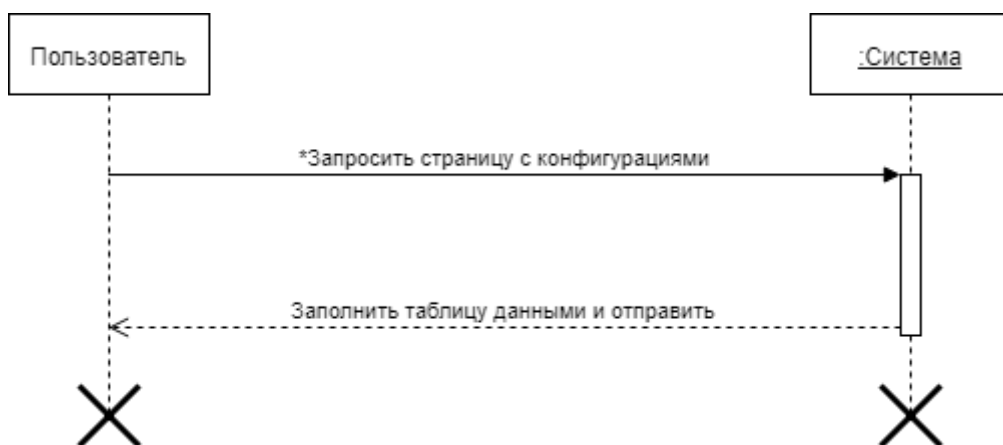
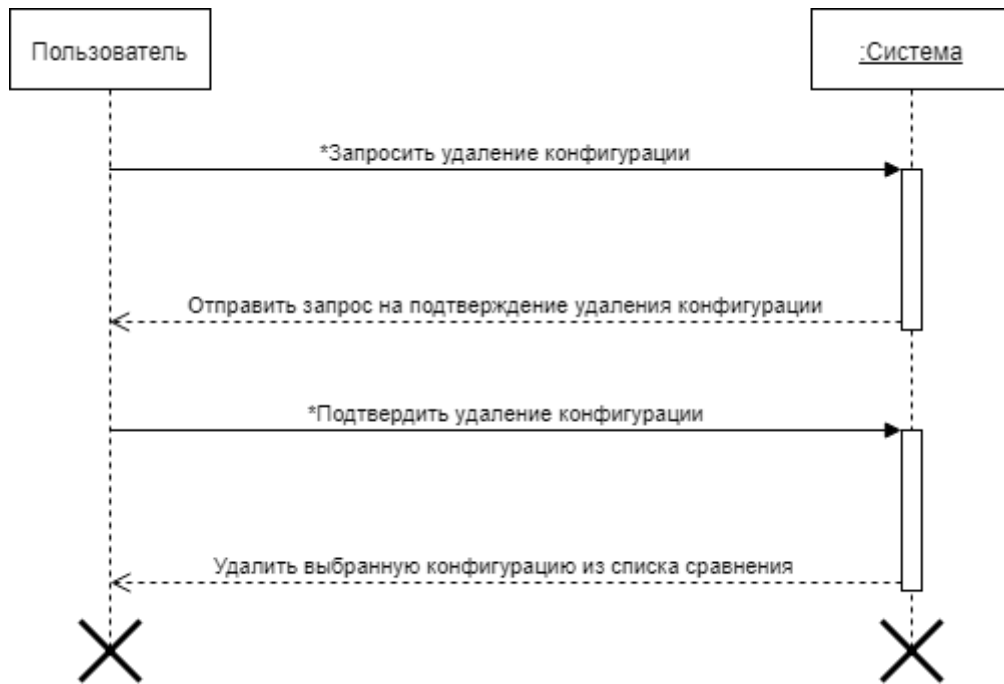


Рисунок 2.13. Сравнение конфигураций

### 2.5.6. Удаление конфигурации

Обмен сообщениями при сравнении конфигураций представлен на рисунке 2.14.



*Рисунок 2.14. Удаление конфигурации*

## **Глава 3. Реализация информационной системы сравнения компьютеров**

После анализа и проектирования основной функционал системы был реализован в виде информационной системы. В данной главе описываются технические средства реализации, представлен интерфейс и тестирование получившейся информационной системы.

### **3.1. Технические средства реализации системы**

Исходя из требования к разрабатываемой системе о том, что система должна реализовывать основной функционал вне зависимости от операционной системы был выбран способ реализации системы в виде веб-приложения. Пользователю достаточно перейти в браузере на нужный адрес, по которому доступен сервер системы, и он получает доступ ко всему функционалу. При этом при определённых настройках функционал системы будет доступен и через всемирную сеть «Интернет» (далее интернет).

Для возможности запуска системы вне зависимости от операционной системы сервера был выбран язык программирования Python [6]. Данный язык является интерпретируемым и для запуска разрабатываемой системы достаточно установить интерпретатор. Поскольку на текущий момент существует две версии (версии 2.7 и 3.6) интерпретатора, которые различаются техническими нюансами – была выбрана версия Python 3.6, как версия с более долгим сроком поддержки сообществом [7]. Сам интерпретатор доступен для использования в коммерческих продуктах без дополнительных выплат, что снимает экономическую нагрузку на пользователя системы.

Для реализации основного функционала системы был выбран каркас Django версии 2.0. Данный каркас позволяет создавать веб-приложения, упрощая для разработчика описание логики работы системы [8]. Данный каркас использует подход выбранного паттерна проектирования MVC. Каркас распространяется под свободной лицензией и не требует дополнительных расходов со стороны пользователя.

В качестве базы данных для системы была выбрана система управления базами данных (далее СКУД) SQLite. Распространяется данная СКУД под общественной лицензией, что также не несёт экономической нагрузки на пользователя [9]. Имеет

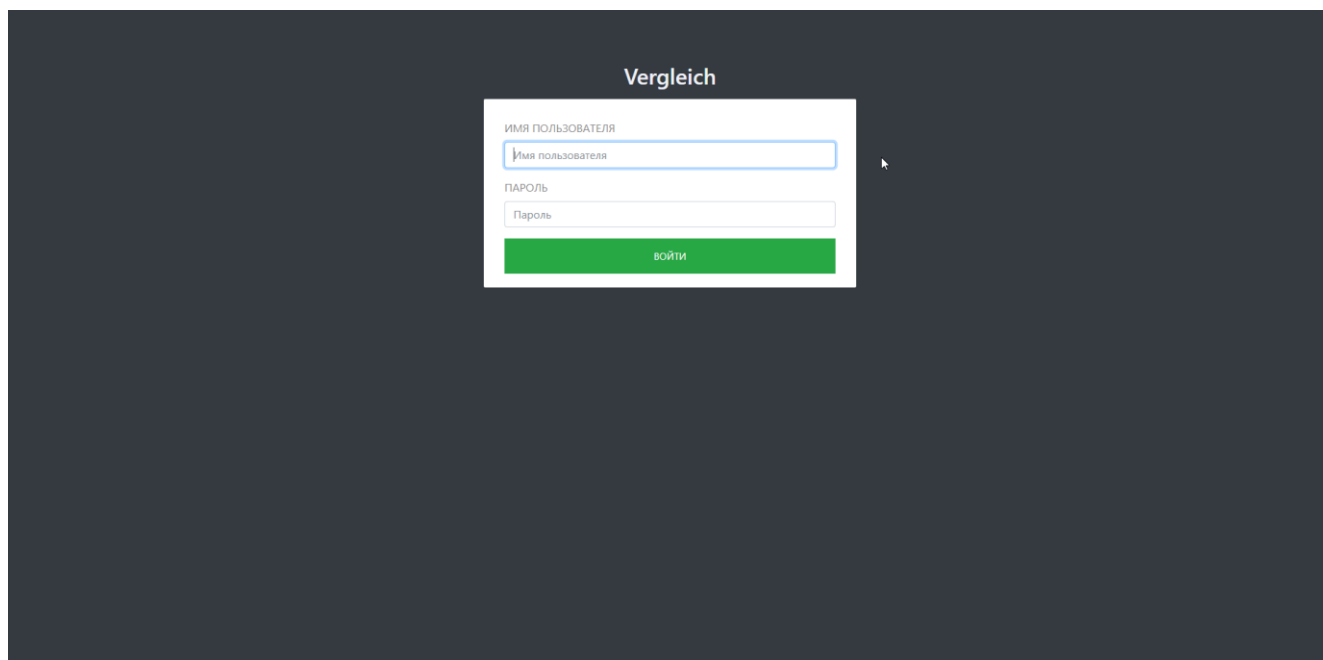
поддержку со стороны Python в виде модуля, включенного в стандартную библиотеку, а значит полностью поддерживается выбранным каркасом для разработки. База данных при этом сохраняется в виде файла, что является удобным для переноса разработанной системы на другие сервера.

### **3.2. Внешний вид разработанной системы**

Информационная система со стороны пользователя является веб-страницами. Для оформления веб-страниц применяют стек технологий HTML-CSS-JS (язык гипертекстовой разметки, каскадные таблицы стилей и интерпретируемый язык программирования JavaScript).

#### **3.2.1. Вход в систему**

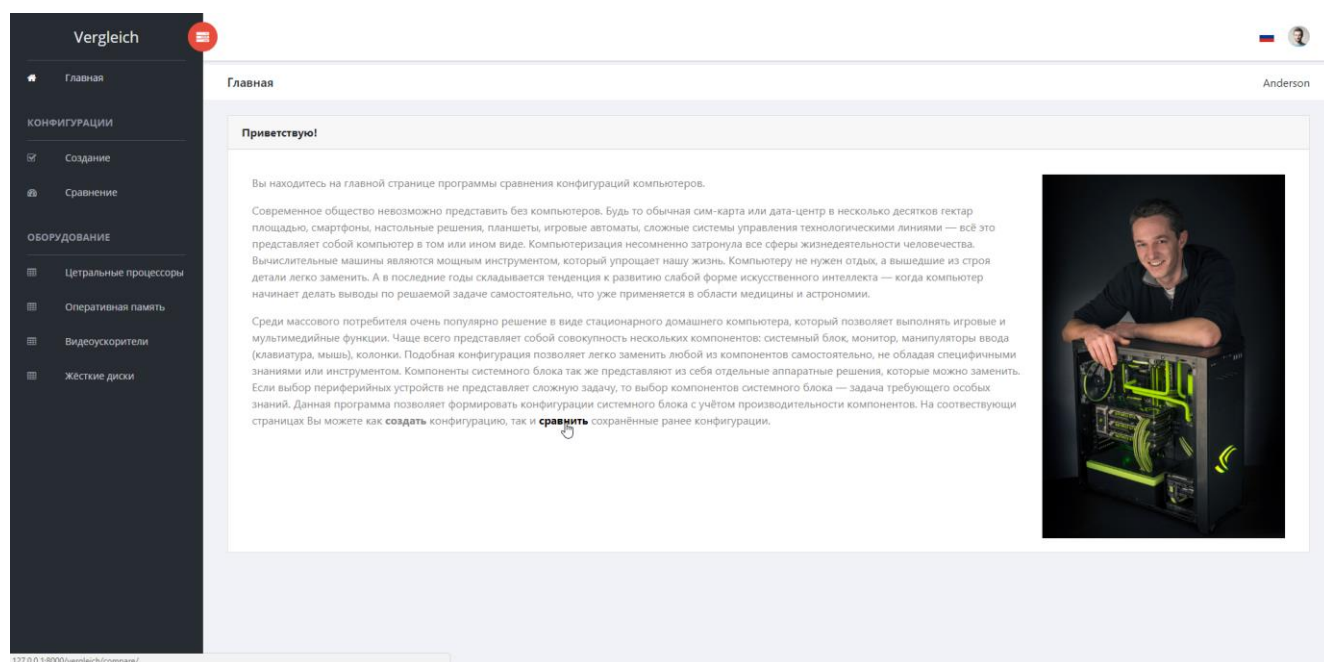
Для входа в систему пользователю предоставляется возможность заполнить форму с полями «Имя пользователя» и «Пароль». При некорректном заполнении поля сбрасываются, приглашая пользователя повторить ввод данных. Снимок экрана формы входа в систему представлен на рисунке 3.1.



***Рисунок 3.1. Форма входа в систему***

На рисунке над формой можно прочесть название системы – Vergleich. Название переводится с немецкого языка как «сравнение».

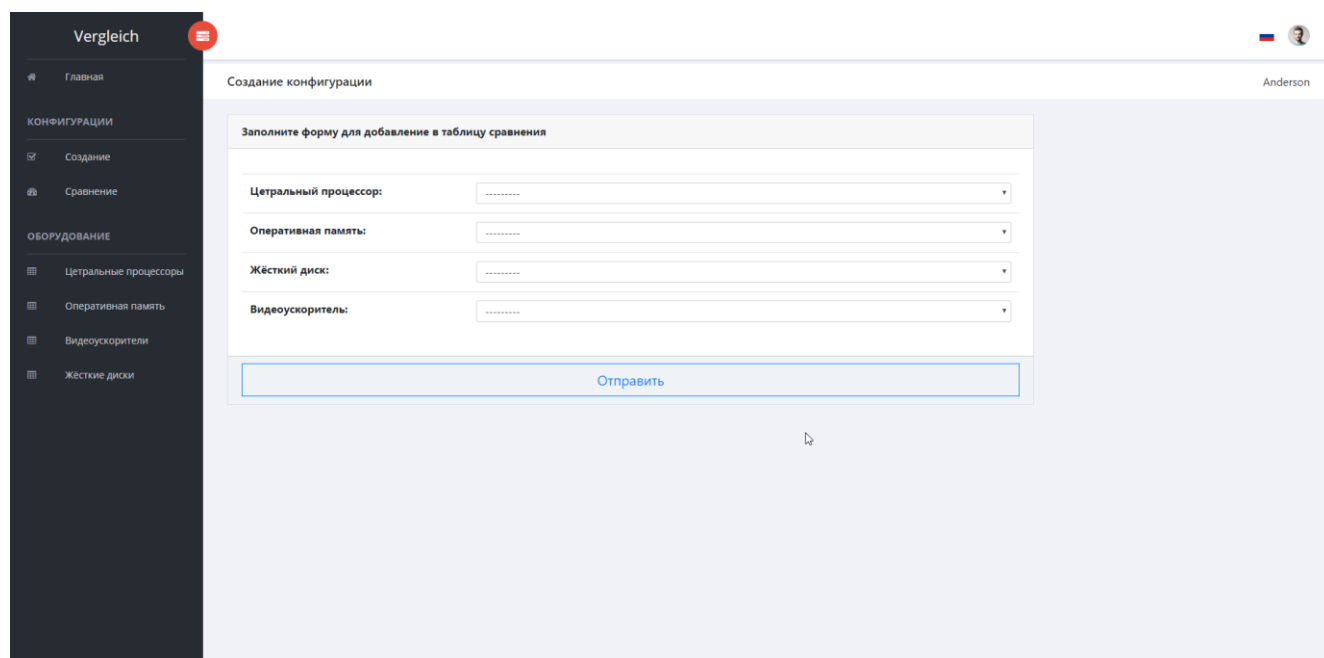
Сразу после выполнения аутентификации пользователь попадает на страницу приветствия. Снимок экрана страницы приветствия представлен на рисунке 3.2.



*Рисунок 3.2. Страница приветствия*

### 3.2.2. Создание конфигурации

Для создания конфигурации для пользователя формируется форма для заполнения. Снимок экрана формы создания конфигурации представлен на рисунке 3.3.



*Рисунок 3.3. Форма создания конфигурации*

### 3.2.3. Сравнение конфигураций

Для сравнения конфигураций сервер подготавливает данные в виде таблицы и отображает её для пользователя. В таблице присутствуют элементы управления, которые позволяют удалять конфигурацию из сравнения или обновить выбранную. Снимок экрана сравнения конфигураций представлен на рисунке 3.4.

	Конф. 1	Конф. 2	Конф. 3	Конф. 4	Конф. 5	Конф. 6
Модель процессора	Intel Core i9-7980XE @ 2.60GHz %	Intel Xeon Gold 6138 @ 2.00GHz %	Intel Core i9-7940X @ 3.10GHz %	Intel Core i9-7960X @ 2.80GHz %	Intel Xeon E5-2696 v3 @ 2.30GHz %	Intel Core i5-4460 @ 3.20GHz %
Счёт	27,817	25,416	25,049	25,656	22,564	6,701
Модель оперативной памяти	A-DATA Technology AM2U139C4P2-8055 4GB %	A-DATA Technology AM2U139C4P2-8055 4GB %	Corsair CMP16GX3M4A1333C9 4GB %	Carry Technology Co. Ltd. U3A8G93-21G9KHAC00 8GB %	Corsair CMD8GX3M2A2133C8 4GB %	Corsair CM3X2G1600C8 2GB %
Скорость чтения	13919	13919	13710	17023	18116	14479
Скорость записи	8340	8340	8648	11416	10499	8700
Задержка	26	26	25	22	21	25
Модель видеоускорителя	Radeon Vega Frontier Edition %	Radeon Vega Frontier Edition %	Quadro K2200M %	GeForce GTX 1050 Ti with Max-Q Design %	GeForce GTX 980 %	GeForce GTX 660 %
Счёт	11,445	11,445	2,614	5,885	9,593	4,127
Модель жёсткого диска	NVMe CX2-8B256-Q11 NV %	Samsung PM961 1TB NVMe %	NVMe LITEON CA1-8D128 %	LITEON LH-256V2G-41 M.2 2260 256GB SED %	Pyro240 %	SAMSUNG MZFLV128HCGR-000MV %
Счёт	10,012	15,189	7,194	8,008	4,937	3,376

Рисунок 3.4. Таблица сравнения конфигураций компьютеров

### 3.2.4. Обновление конфигурации

Для обновления конфигурации пользователю необходимо произвести щелчок на управляющем элементе в последней строке таблицы в виде карандаша. При этом сформируется форма как при создании конфигурации, но с управляющим элементом отправки формы. Снимок экрана обновления конфигурации представлен на рисунке ниже (см. рис. 3.5).

### 3.2.5. Поиск единицы оборудования в базе данных

Поиск единицы оборудования в базе данных в базе данных для любого типа оборудования выполнен в виде таблицы с формой для ввода критериев фильтрации. При этом для пользователя доступны следующие управляющие элементы:

Заголовки столбцов являются активными элементами и щелчок по заголовку отсортирует таблицу по данному столбцу. Один щелчок — значения по убыванию. Второй щелчок — значения по возрастанию.

Vergleich

Главная

КОНФИГУРАЦИИ

Создание

Сравнение

ОБОРУДОВАНИЕ

Центральные процессоры

Оперативная память

Видеоускорители

Жесткие диски

Создание конфигурации

Admin

Измените компоненты и сохраните конфигурацию

Центральный процессор:

Intel Xeon Gold 6138 @ 2.00GHz

Оперативная память:

A-DATA Technology AM2U139CAP2-8055 4GB

Видеоускорители:

Radeon Vega Frontier Edition

Жесткий диск:

Samsung PM961 1TB NVMe

Отправить

**Рисунок 3.5 Форма обновления конфигурации**

Vergleich

Главная

КОНФИГУРАЦИИ

Создание

Сравнение

ОБОРУДОВАНИЕ

Центральные процессоры

Оперативная память

Видеоускорители

Жесткие диски

Сводная таблица

Anderson

Центральные процессоры

Имя	Очки	Позиция	В наличии	Цена
Intel Core i9-7980XE @ 2.60GHz	27,817	2	<input checked="" type="checkbox"/>	1890,87
Intel Xeon Gold 6154 @ 3.00GHz	27,789	3	<input checked="" type="checkbox"/>	3543,0
Intel Xeon W-2195 @ 2.30GHz	26,47	4	<input checked="" type="checkbox"/>	2730,26
Intel Core i9-7960X @ 2.80GHz	25,656	5	<input checked="" type="checkbox"/>	1564,04
Intel Xeon Gold 6138 @ 2.00GHz	25,416	6	<input checked="" type="checkbox"/>	2689,99
Intel Xeon E5-2679 v4 @ 2.50GHz	25,236	7	<input checked="" type="checkbox"/>	2702,0
Intel Core i9-7940X @ 3.10GHz	25,049	8	<input checked="" type="checkbox"/>	1366,93
Intel Xeon E5-2699 v4 @ 2.20GHz	23,392	10	<input checked="" type="checkbox"/>	8010,0
Intel Core i9-7920X @ 2.90GHz	23,185	11	<input checked="" type="checkbox"/>	1101,01
Intel Xeon E5-2696 v3 @ 2.30GHz	22,564	12	<input checked="" type="checkbox"/>	2498,0

Страница 1 из 67Следующая

Фильтр

Имя содержит

Счёт между

Начиная с ...

... до

Позиция

Наличие

Неизвестно

Цена между

Начиная с ...

... до

Отправить

**Рисунок 3.6 Таблица поиска единицы оборудования в базе данных**

40



Внизу таблицы располагаются элементы перехода на «страницам» таблицы. По умолчанию для пользователя предоставляется не вся таблица, а только часть для удобства просмотра. Пользователь может перейти на следующие «страницы» щелчком по соответствующему активному элементу.

Сбоку таблицы располагается форма для задания критериев фильтрации отображаемых данных. Пользователь отправляет форму нажатием кнопки, расположенной внизу формы.

Снимок экрана поиска единицы оборудования в базе данных представлен на рисунке выше (см. рис. 3.6).

### **3.3. Тестирование информационной системы**

Для того чтобы удостовериться в надёжности созданной системы необходимо провести её тестирование. Возможны два варианта тестирования: ручное и автоматизированное.

Ручное тестирование возможно при проработке основного функционала, связанного с графическим интерфейсом системы. С ростом возможностей системы данный тип тестирования становится всё затруднительнее, поскольку человек может банально устать, а также возможны ошибки по невнимательности. В разрабатываемой системе было проведено ручное тестирование всех основных функциональных узлов в рамках графического интерфейса.

Автоматическое тестирование позволяет проверить систему более глубоко: тестирование внутренней логики отдельных компонентов. Автоматическое тестирование гораздо быстрее ручного, но требует большего времени для разработки системы в целом в связи с тем, что тесты сами являются программами. Данный тип тестирования может оперировать значительно большим объёмом входных данных.

Согласно документации на каркас Django – необходимо подвергать тестированию только то, что написал сам разработчик [8]. В связи с этим было решено написать модульные тесты, которые направлены на тестирование конкретных методов, написанных при разработке.

Для возможности количественно оценить результаты покрытия написанной системы тестами был задействован модель «coverage». С помощью специальных команд данный модуль проверяет программу на языке Python и после проверки позволяет

выгрузить в формате гипертекстовой разметки отчёт [10]. В отчёте есть возможность увидеть те места программы, которые либо уже покрыты тестами, либо те, которые ещё предстоит ими покрыть. Пример подсветки не охваченных тестами функций программы представлен на рисунке 3.7.

```
55 # Video table
56 class ScrapperBenchVideo(models.Model):
57     name = models.CharField(max_length=1000, blank=True)
58     url = models.URLField(max_length=1000, blank=True)
59     score = models.FloatField(blank=True)
60     rank = models.IntegerField(blank=True)
61     in_stock = models.BooleanField(default=False)
62     price = models.FloatField(blank=True)
63
64     def __str__(self):
65         return self.name
66
67     class Meta:
68         ordering = ['name']
```


**Рисунок 3.7. Подсветка функций, не охваченных тестами, сгенерированная модулем coverage**

После написания теста на подсвеченную функцию – подсветка пропадает. Это видно на рисунке 3.8.

```
55 # Video table
56 class ScrapperBenchVideo(models.Model):
57     name = models.CharField(max_length=1000, blank=True)
58     url = models.URLField(max_length=1000, blank=True)
59     score = models.FloatField(blank=True)
60     rank = models.IntegerField(blank=True)
61     in_stock = models.BooleanField(default=False)
62     price = models.FloatField(blank=True)
63
64     def __str__(self):
65         return self.name
66
67     class Meta:
68         ordering = ['name']
69
```

**Рисунок 3.8. Снятие подсветки с функций, покрытых тестами, сгенерированная модулем coverage**

Были написаны тесты, которые покрывают все написанные функции. Пример отчёта представлен на рисунке ниже (см. рис. 3.9).

Coverage report: 100%		<input type="text" value="filter..."/>		
Module ↓	statements	missing	excluded	coverage
vergleich\filters.py	30	0	0	100%
vergleich\forms.py	6	0	0	100%
vergleich\models.py	58	0	0	100%
vergleich\tables.py	57	0	0	100%
vergleich\views.py	120	0	0	100%
<b>Total</b>	<b>521</b>	<b>0</b>	<b>0</b>	<b>100%</b>
coverage.py v4.5.1, created at 2018-05-22 15:32				

*Рисунок 3.9. Оценка покрытия написанных функций тестами, сгенерированная с помощью модуля coverage*

Исходный код тестов приведён в приложении Б.

## **Заключение**

Разработанная информационная система сравнения компьютеров решает поставленную цель. Для формирования требований к системе были проанализированы различные источники информации, а также системы с аналогичным функционалом. В процессе проектирования системы были созданы четыре типа диаграмм:

- Диаграмма прецедентов
- Диаграммы активностей
- Диаграммы классов
- Диаграммы последовательностей

При разработке информационной системы были написаны 1461 строчка исходного кода, в которые входят 40 классов. Разработан удобный интерфейс пользователя, который в полной мере покрывает требования к системе.

В процессе работы над проектом так же была достигнута цель в углублении имеющихся знаний по разработке информационных систем, улучшены навыки работы с базами данных, на практике были испытаны новые методы нахождения и анализа важной информации, а также получены приятные эмоции при технической реализации системы.

## Библиографический список

- [1] Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. — СПб.: Питер, 2015. — 1120 с.
- [2] Мураховский В. И. Устройство компьютера. АСТ-Пресс Книга, 2003 – 640 с.
- [3] Интуит. Национальный открытый университет [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.intuit.ru/studies/courses/1007/229/lecture/5950>
- [4] Бэрри, Пол. Изучаем программирование на Python / Пол Бэрри; [пер. с англ. М.А. Райтман]. – Москва: Издательство «Э», 2017. – 624 с.
- [5] Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2015. – 368 с.
- [6] Макконнелл С. Совершенный код. Мастер-класс / Пер. с англ – М.: Издательство «Русская редакция», 2010. – 896 стр.: ил.
- [7] Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с., ил.
- [8] Django documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://docs.djangoproject.com/en/2.0/>
- [9] SQLite Home Page [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.sqlite.org/index.html>
- [10] Coverage.py [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://coverage.readthedocs.io/en/coverage-4.5.1/>

## Приложение А. Диаграммы классов

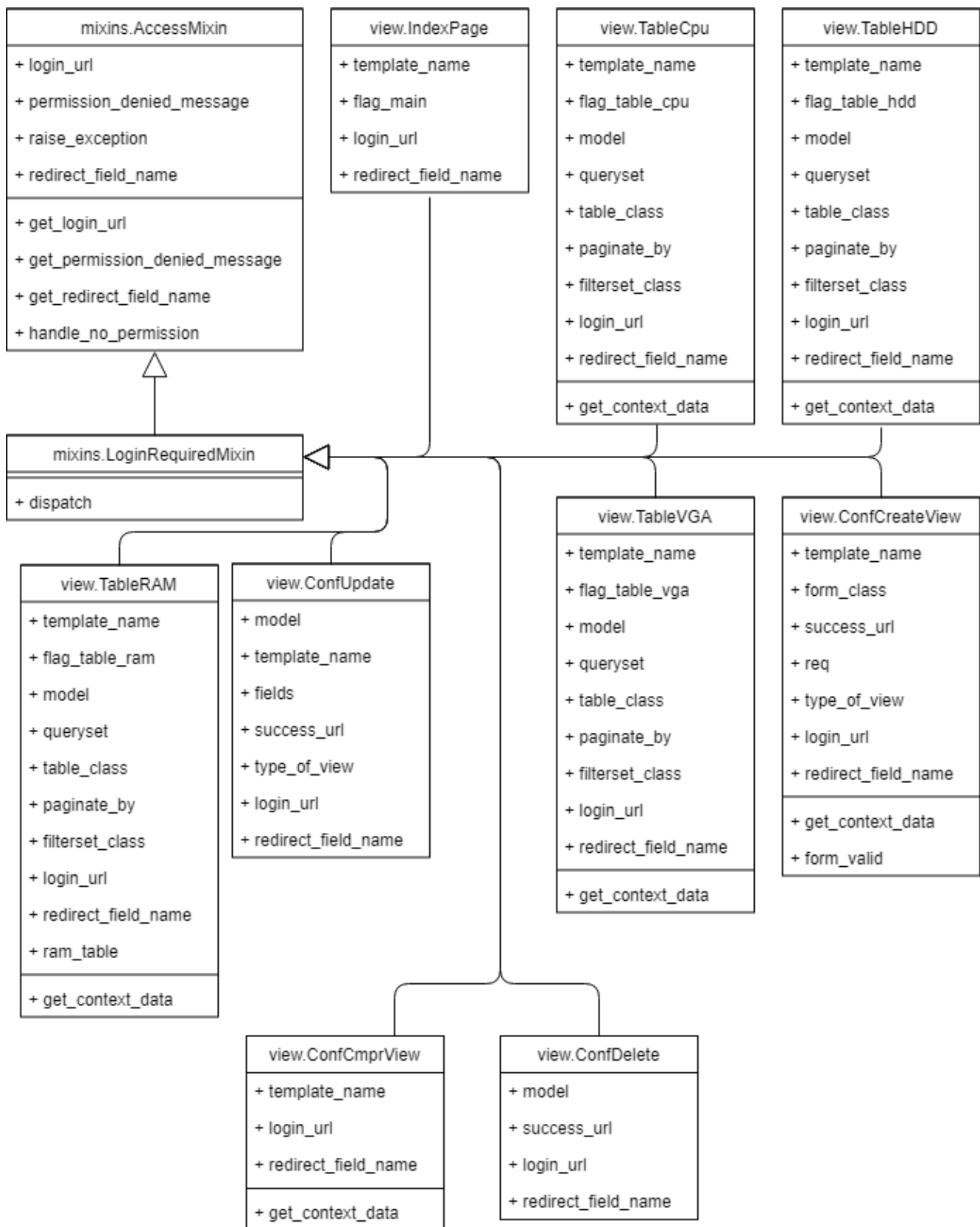
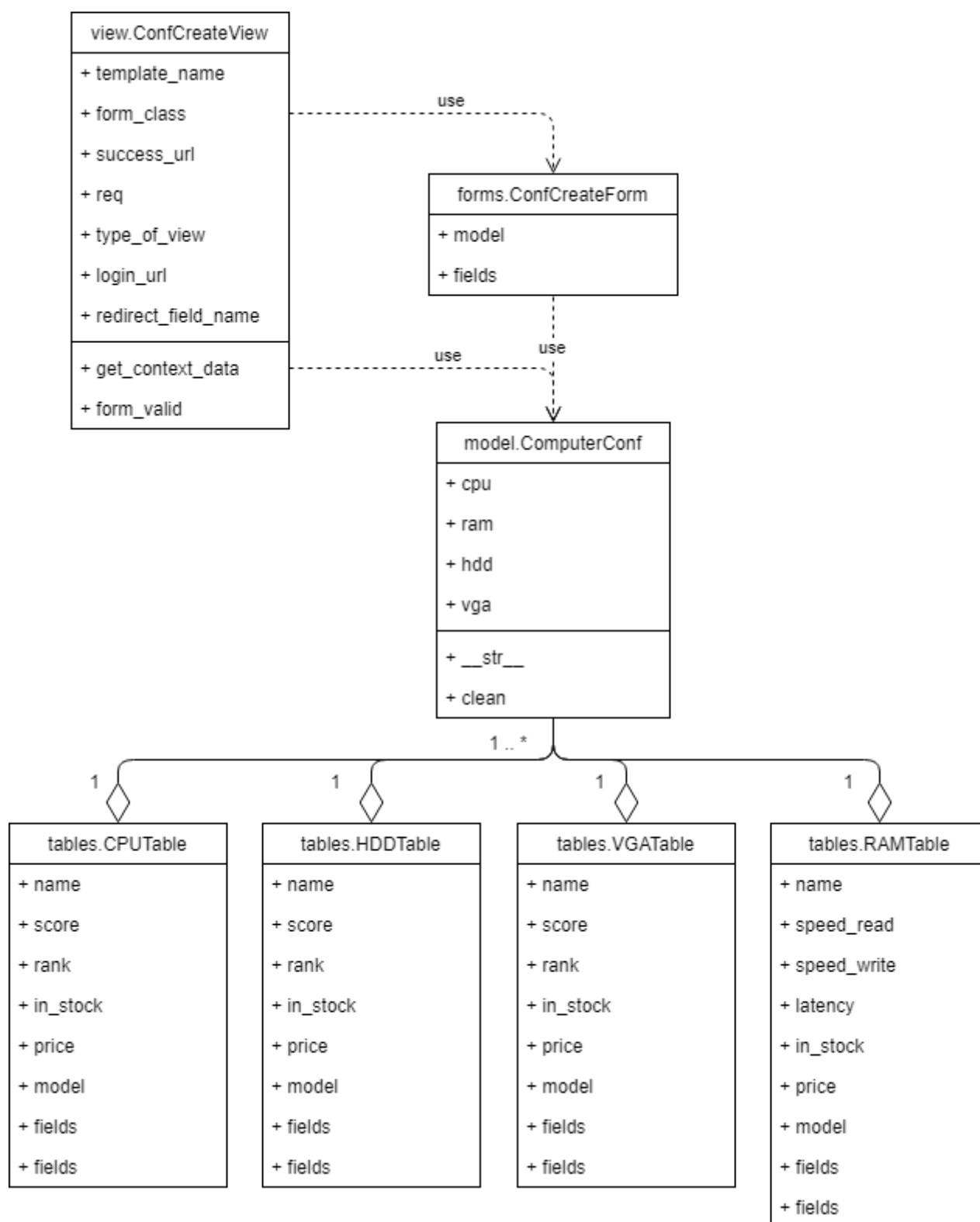
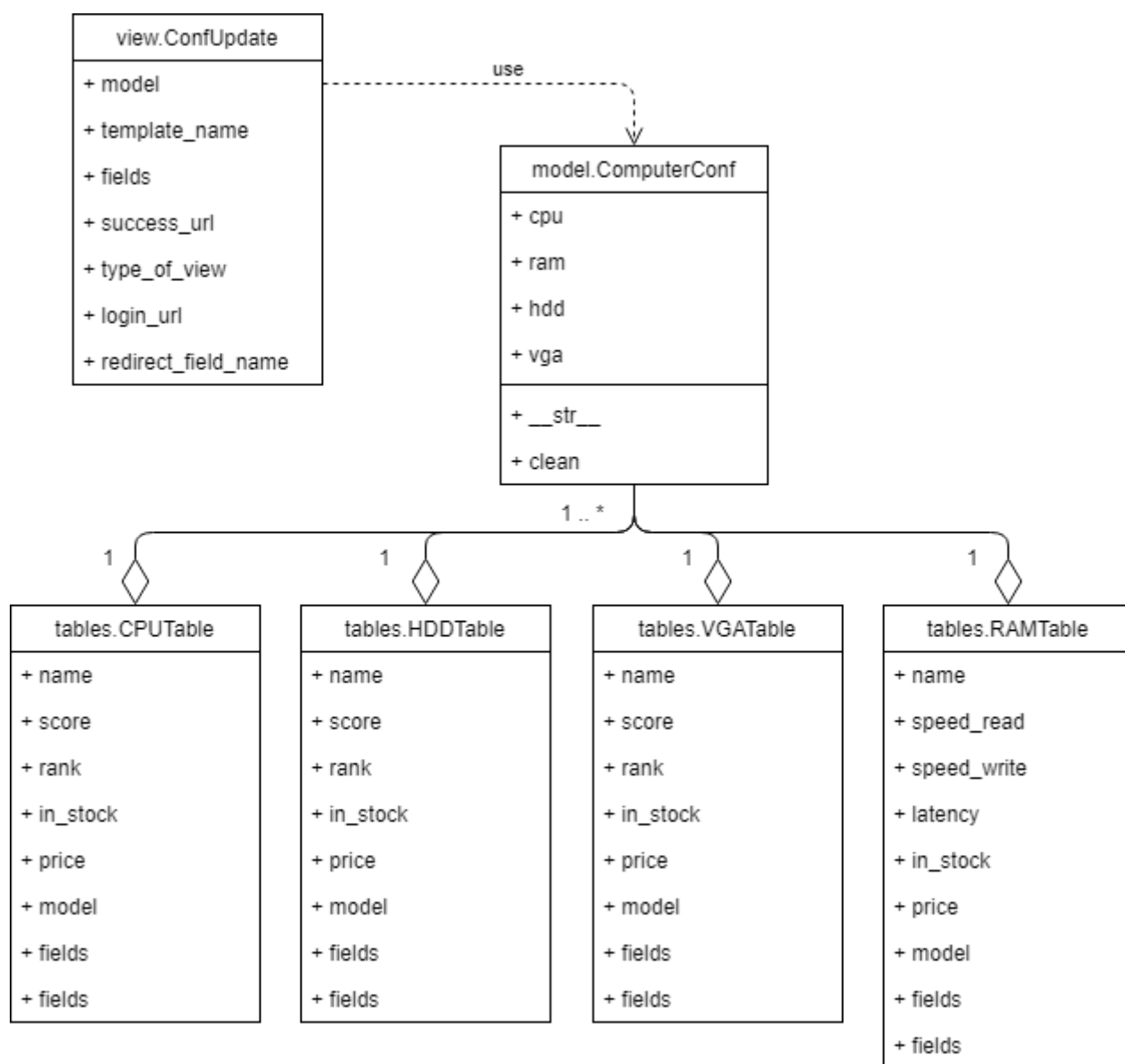


Рисунок А.1. Диаграмма классов «Вход в систему»

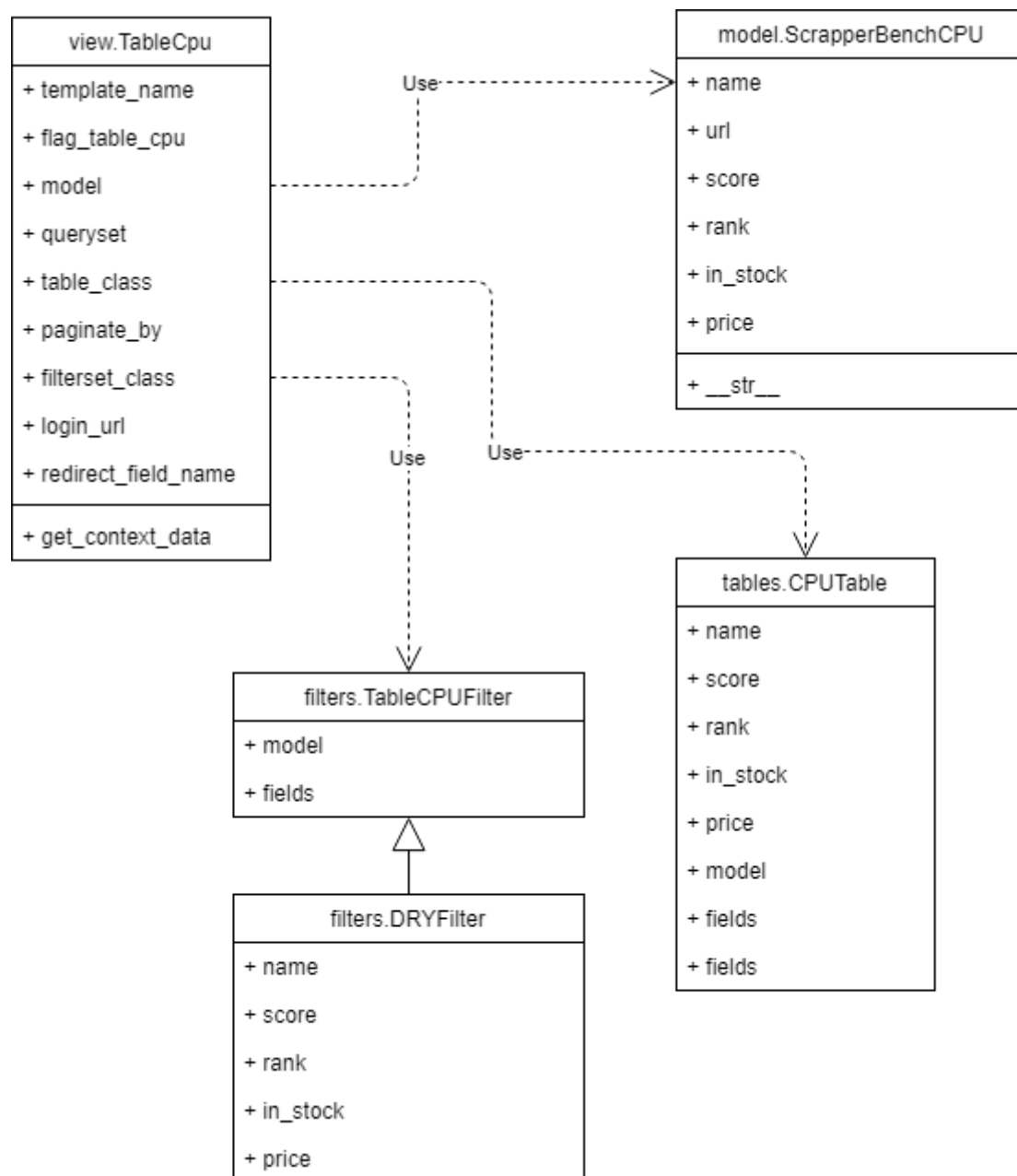


**Рисунок А.2. Диаграмма классов «Конструирование конфигурации»**

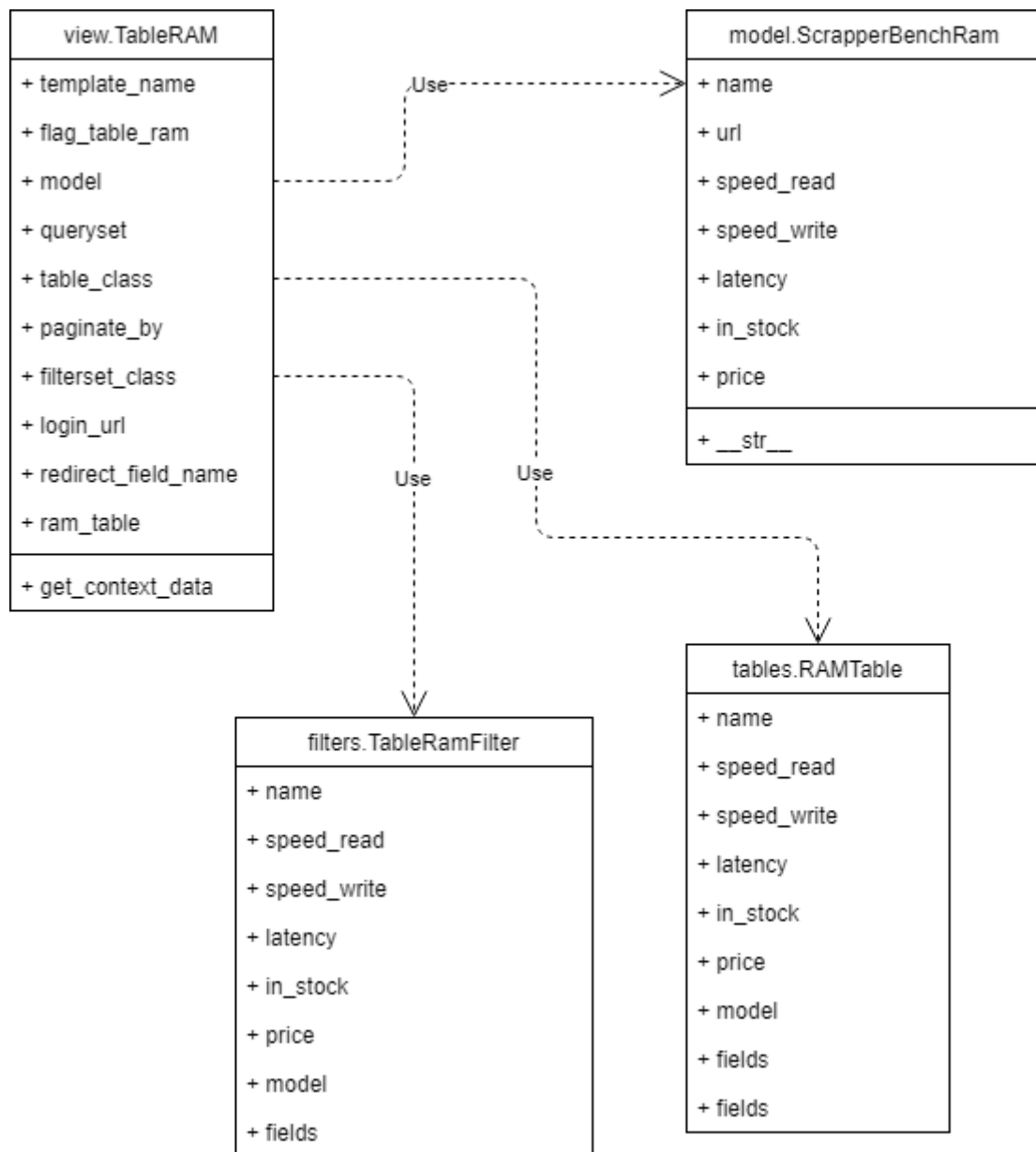


*Рисунок А.3. Диаграмма классов «Обновление конфигурации»*

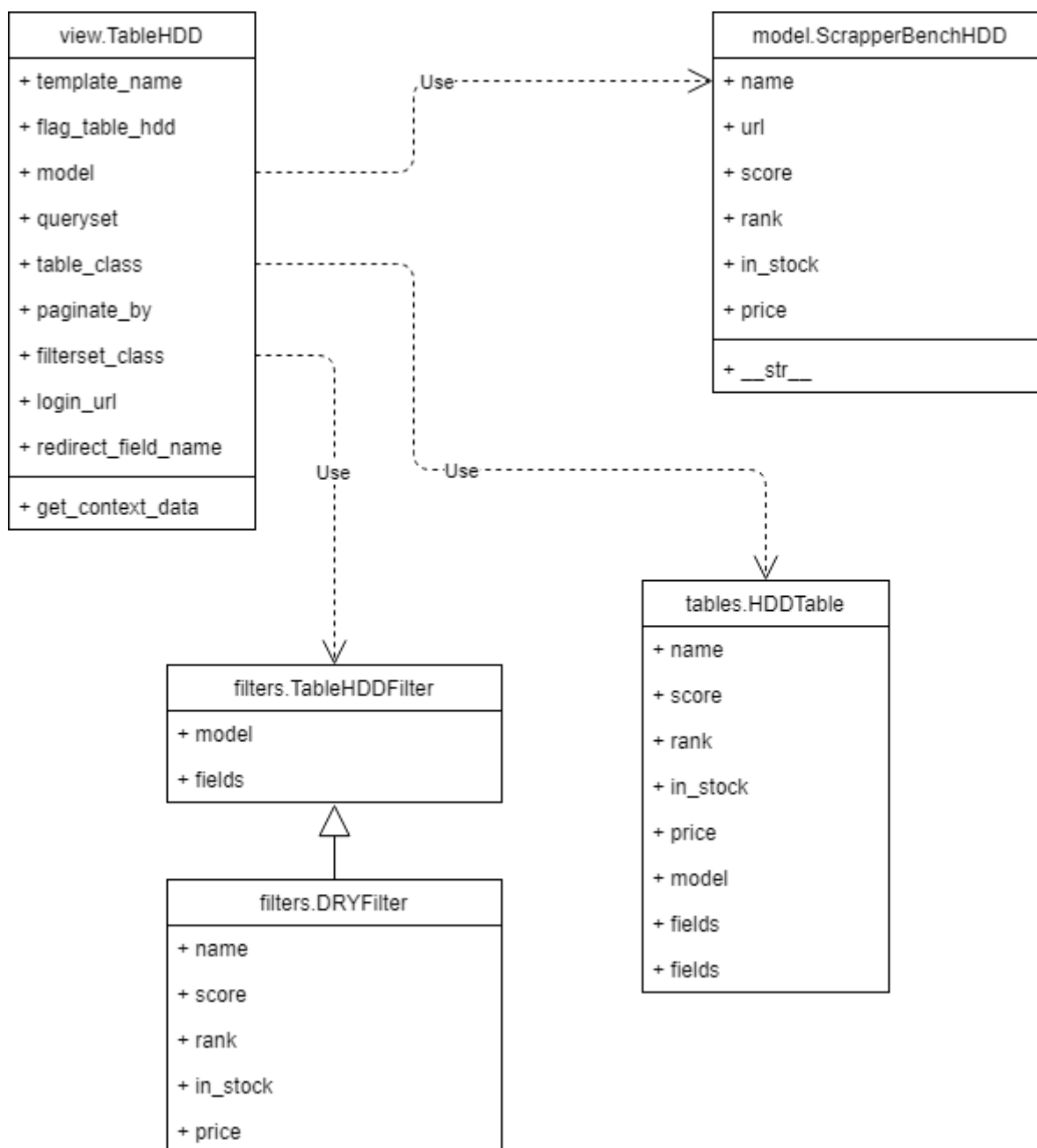




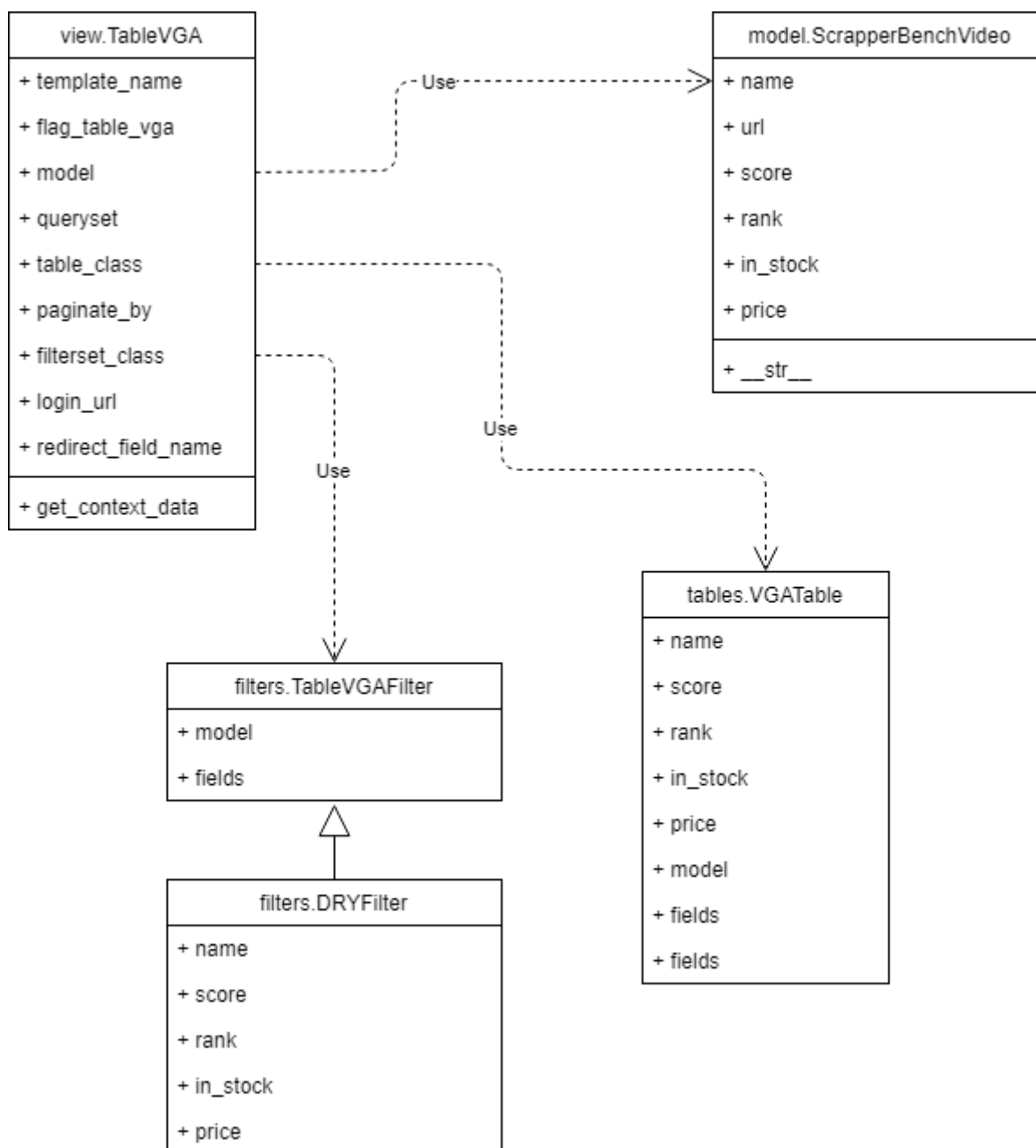
**Рисунок А.4. Диаграмма классов «Поиск в БД проекта моделей центральных процессоров»**



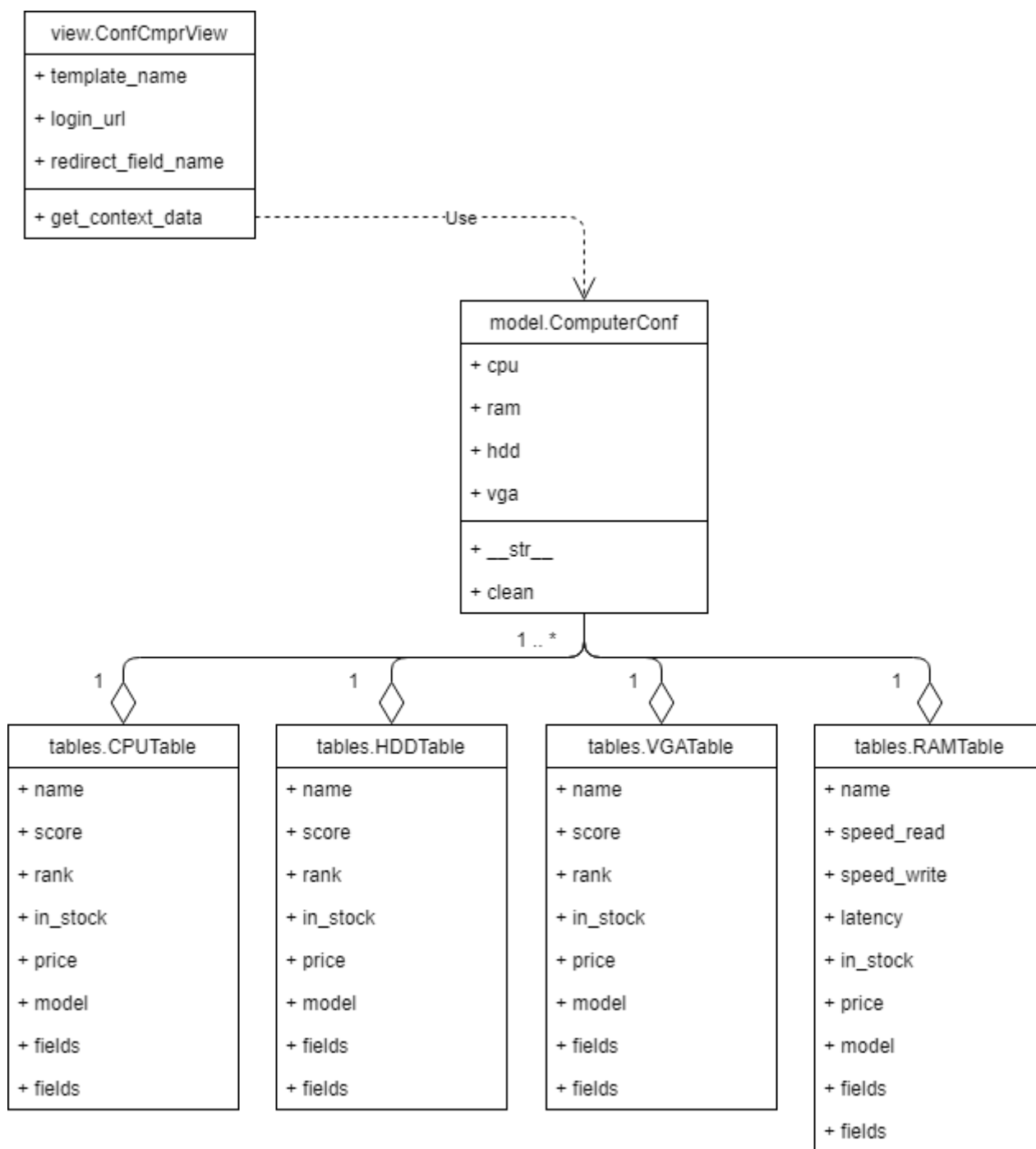
*Рисунок А.5. Диаграмма классов «Поиск в БД проекта моделей оперативной памяти»*



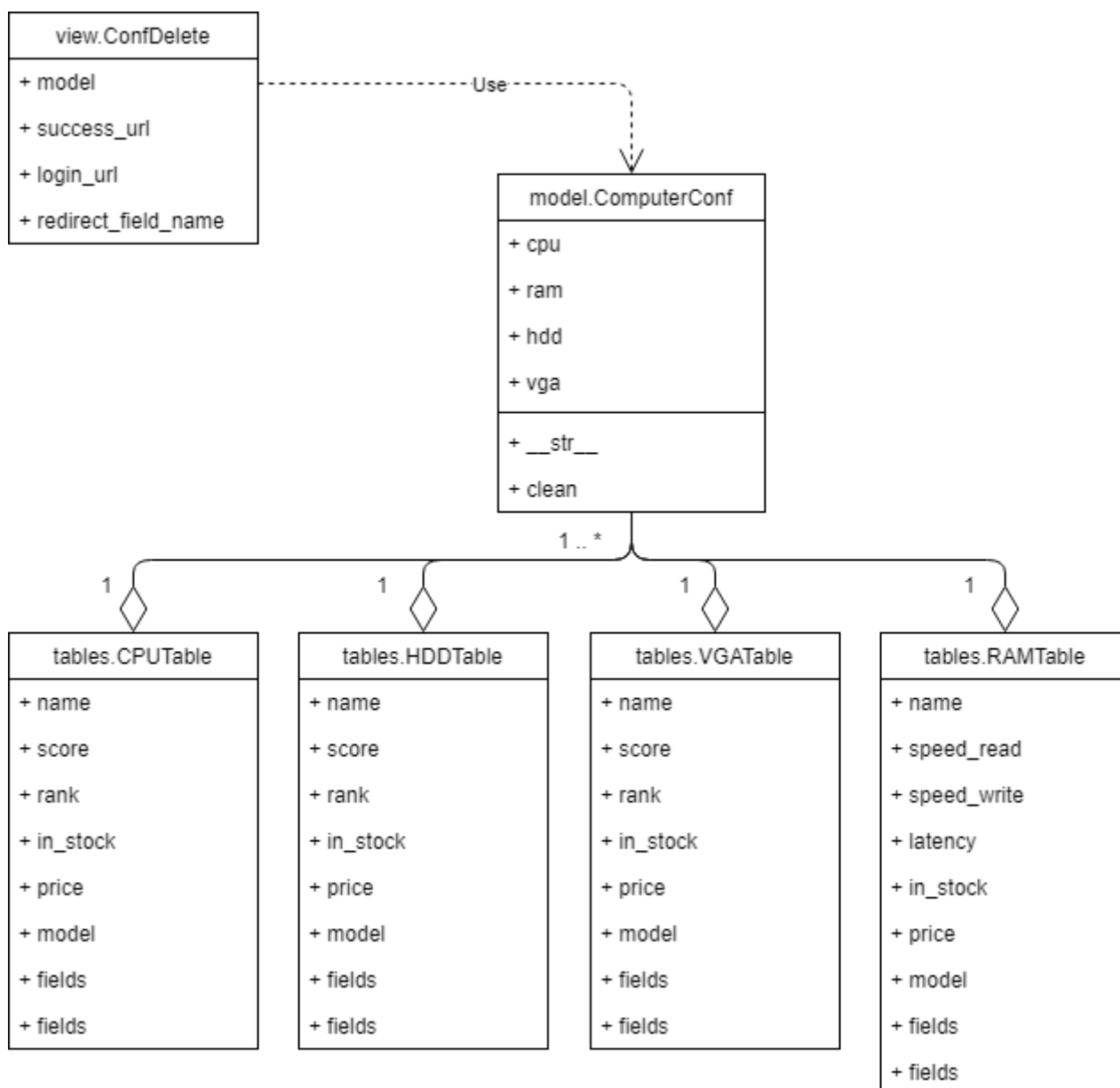
**Рисунок А.6. Диаграмма классов «Поиск в БД проекта моделей жёстких дисков»**



**Рисунок А.7. Диаграмма классов «Поиск в БД проекта моделей видеоускорителей»**



*Рисунок А.8. Диаграмма классов «Сравнение конфигураций»*



**Рисунок А.9. Диаграмма классов «Удаление конфигурации»**

## Приложение Б. Исходный код модульных тестов.

В приложении приведен исходный код модульных тестов разработанной информационной системы. Поскольку код разбит по модулям – в приложении так же производится разбиение по модулям.

### Б.1. Модуль test\_form.py

```
from django.test import TestCase

from vergleich import forms, models

class ConfCmprViewTest(TestCase):

    def setUp(self):
        models.ScrapperBenchRam.objects.create(
            name = "RamName",
            url = "someRAM.url",
            speed_read = 1598,
            speed_write = 8951,
            latency = 2587,
            in_stock = True,
            price = 999999
        )
        models.ScrapperBenchVideo.objects.create(
            name = "VgaName",
            url = "someVGA.url",
            score = 99999,
            rank = -1,
            in_stock = True,
            price = 999999
        )
        models.ScrapperBenchHDD.objects.create(
            name = "HddName",
            url = "someHDD.url",
            score = 99999,
            rank = -1,
            in_stock = True,
            price = 999999
        )
```

```

models.ScrapperBenchCPU.objects.create(
    name = "CpuName",
    url = "someCPU.url",
    score = 99999,
    rank = -1,
    in_stock = True,
    price = 999999
)

models.ComputerConf.objects.create(
    cpu = models.ScrapperBenchCPU.objects.get(id=1),
    ram = models.ScrapperBenchRam.objects.get(id=1),
    hdd = models.ScrapperBenchHDD.objects.get(id=1),
    vga = models.ScrapperBenchVideo.objects.get(id=1)
)

def test_forms(self):
    test_date = models.ComputerConf(
        cpu = models.ScrapperBenchCPU.objects.all().first(),
        ram = models.ScrapperBenchRam.objects.all().first(),
        hdd = models.ScrapperBenchHDD.objects.all().first(),
        vga = models.ScrapperBenchVideo.objects.all().first()
    )
    test_form = forms.ConfCreateForm(instance=test_date)
    form_data = {
        'cpu': models.ScrapperBenchCPU.objects.all().first().id,
        'ram': models.ScrapperBenchRam.objects.all().first().id,
        'hdd': models.ScrapperBenchHDD.objects.all().first().id,
        'vga': models.ScrapperBenchVideo.objects.all().first().id
    }
    test_form = forms.ConfCreateForm(form_data, instance=test_date)
    self.assertTrue(test_form.is_valid())

```

## Б.2. Модуль test\_models.py

```

from django.core.exceptions import ValidationError
from django.test import TestCase

from vergleich.models import (ComputerConf, ScrapperBenchCPU, ScrapperBenchHDD,
                             ScrapperBenchRam, ScrapperBenchVideo)

class CpuModelTest(TestCase):

    """Unit tests for CPU model"""

```



```

def setUp(self):
    ScrapperBenchCPU.objects.create(
        name = "Some name of CPU model for test",
        url = "someCPU.url",
        score = 99999,
        rank = -1,
        in_stock = True,
        price = 999999
    )

def test_cpu_name_max_length(self):
    pos = ScrapperBenchCPU.objects.get(id=1)
    max_length = pos._meta.get_field('name').max_length
    self.assertEqual(max_length, 1000)

def test_cpu_url_max_length(self):
    pos = ScrapperBenchCPU.objects.get(id=1)
    max_length = pos._meta.get_field('url').max_length
    self.assertEqual(max_length, 1000)

def test_cpu_model_str_method(self):
    obj = ScrapperBenchCPU.objects.get(id=1)
    self.assertEqual(str(obj), obj.name)

class HDDModelTest(TestCase):

    """Unit tests for HDD model"""

    def setUp(self):
        ScrapperBenchHDD.objects.create(
            name = "Some name of HDD model for test",
            url = "someHDD.url",
            score = 99999,
            rank = -1,
            in_stock = True,
            price = 999999
        )

    def test_hdd_name_max_length(self):
        pos = ScrapperBenchHDD.objects.get(id=1)
        max_length = pos._meta.get_field('name').max_length
        self.assertEqual(max_length, 1000)

    def test_hdd_url_max_length(self):
        pos = ScrapperBenchHDD.objects.get(id=1)
        max_length = pos._meta.get_field('url').max_length
        self.assertEqual(max_length, 1000)

    def test_hdd_model_str_method(self):
        obj = ScrapperBenchHDD.objects.get(id=1)
        self.assertEqual(str(obj), obj.name)

class VGAModelTest(TestCase):

    """Unit tests for VGA model"""

    def setUp(self):
        ScrapperBenchVideo.objects.create(
            name = "Some name of VGA model for test",
            url = "someVGA.url",
            score = 99999,

```

```

        rank = -1,
        in_stock = True,
        price = 999999
    )

def test_vga_name_max_length(self):
    pos = ScrapperBenchVideo.objects.get(id=1)
    max_length = pos._meta.get_field('name').max_length
    self.assertEqual(max_length, 1000)

def test_vga_url_max_length(self):
    pos = ScrapperBenchVideo.objects.get(id=1)
    max_length = pos._meta.get_field('url').max_length
    self.assertEqual(max_length, 1000)

def test_vga_model_str_method(self):
    obj = ScrapperBenchVideo.objects.get(id=1)
    self.assertEqual(str(obj), obj.name)

class RAMModelTest(TestCase):

    """Unit tests for RAM model"""

    def setUp(self):
        ScrapperBenchRam.objects.create(
            name = "Some name of RAM model for test",
            url = "someRAM.url",
            speed_read = 1598,
            speed_write = 8951,
            latency = 2587,
            in_stock = True,
            price = 999999
        )

    def test_ram_name_max_length(self):
        pos = ScrapperBenchRam.objects.get(id=1)
        max_length = pos._meta.get_field('name').max_length
        self.assertEqual(max_length, 1000)

    def test_ram_url_max_length(self):
        pos = ScrapperBenchRam.objects.get(id=1)
        max_length = pos._meta.get_field('url').max_length
        self.assertEqual(max_length, 1000)

    def test_ram_model_str_method(self):
        obj = ScrapperBenchRam.objects.get(id=1)
        self.assertEqual(str(obj), obj.name)

class ComputerConfModelTest(TestCase):

    def setUp(self):
        ScrapperBenchRam.objects.create(
            name = "Some name of RAM model for test",
            url = "someRAM.url",
            speed_read = 1598,
            speed_write = 8951,
            latency = 2587,
            in_stock = True,
            price = 999999
        )
        ScrapperBenchVideo.objects.create(

```

```

        name = "Some name of VGA model for test",
        url = "someVGA.url",
        score = 99999,
        rank = -1,
        in_stock = True,
        price = 999999
    )
    ScrapperBenchHDD.objects.create(
        name = "Some name of HDD model for test",
        url = "someHDD.url",
        score = 99999,
        rank = -1,
        in_stock = True,
        price = 999999
    )
    ScrapperBenchCPU.objects.create(
        name = "Some name of CPU model for test",
        url = "someCPU.url",
        score = 99999,
        rank = -1,
        in_stock = True,
        price = 999999
    )
    for i in range(0, 8, 1):
        ComputerConf.objects.create(
            cpu = ScrapperBenchCPU.objects.get(id=1),
            ram = ScrapperBenchRam.objects.get(id=1),
            hdd = ScrapperBenchHDD.objects.get(id=1),
            vga = ScrapperBenchVideo.objects.get(id=1)
        )

    def test_computerconf_model_str_method(self):
        obj = ComputerConf.objects.get(id=1)
        self.assertEqual(str(obj), "{0}; {1}; {2}; {3}".format(obj.cpu, obj.ram,
obj.hdd, obj.vga))

    def test_computerconf_model_clean_method(self):
        inst = ComputerConf()
        self.assertRaises(ValidationError, inst.clean)

```

### Б.3. Модуль test\_view.py

```

from django.contrib.auth.models import User
from django.test import Client, TestCase
from django.urls import reverse

from test_plus.test import CBVTestCase
from vergleich import filters, forms, models
from vergleich.views import (ConfCmprView, ConfCreateView, TableCpu, TableHDD,
                             TableRAM, TableVGA)

class LogInTest(TestCase):
    def setUp(self):
        self.credentials = {
            'username': 'testuser',
            'password': 'secret'}
        User.objects.create_user(**self.credentials)
    def test_login(self):

```

```

        # send login data
        response = self.client.post('/accounts/login/', self.credentials,
follow=True)
        # should be logged in now
        self.assertTrue(response.context['user'].is_authenticated)

class TableCpuTest(TestCase):

    def setUp(self):
        user = User.objects.create_user('temporary', 'temporary@gmail.com',
'temporary')

    def test_tablecpu_view_responce(self):
        self.client.login(username='temporary', password='temporary')
        resp = self.client.get(reverse('table_cpu'), follow=True)
        self.assertEqual(resp.status_code, 200)

    def test_tablecpu_view_page_context(self):
        self.client.login(username='temporary', password='temporary')
        resp = self.client.get(reverse('table_cpu'), follow=True)
        self.assertEqual(resp.context['page_name'], "Центральные процессоры")
        self.assertEqual(resp.context['filter'], filters.TableCPUFilter)

class TableHDDTest(TestCase):

    def setUp(self):
        user = User.objects.create_user('temporary', 'temporary@gmail.com',
'temporary')

    def test_tablehdd_view_responce(self):
        self.client.login(username='temporary', password='temporary')
        resp = self.client.get(reverse('table_hdd'), follow=True)
        self.assertEqual(resp.status_code, 200)

    def test_tablehdd_view_page_context(self):
        self.client.login(username='temporary', password='temporary')
        resp = self.client.get(reverse('table_hdd'), follow=True)
        self.assertEqual(resp.context['page_name'], "Жёсткие диски")
        self.assertEqual(resp.context['filter'], filters.TableHDDFilter)

class TableVGATest(TestCase):

    def setUp(self):
        user = User.objects.create_user('temporary', 'temporary@gmail.com',
'temporary')

    def test_tablevga_view_responce(self):
        self.client.login(username='temporary', password='temporary')
        resp = self.client.get(reverse('table_vga'), follow=True)
        self.assertEqual(resp.status_code, 200)

    def test_tablevga_view_page_context(self):
        self.client.login(username='temporary', password='temporary')
        resp = self.client.get(reverse('table_vga'), follow=True)
        self.assertEqual(resp.context['page_name'], "Видеоускорители")
        self.assertEqual(resp.context['filter'], filters.TableVGAFilter)

class TableRAMTest(TestCase):

```

```

def setUp(self):
    user = User.objects.create_user('temporary', 'temporary@gmail.com',
'temporary')

def test_tablevga_view_responce(self):
    self.client.login(username='temporary', password='temporary')
    resp = self.client.get(reverse('table_ram'), follow=True)
    self.assertEqual(resp.status_code, 200)

def test_tablevga_view_page_context(self):
    self.client.login(username='temporary', password='temporary')
    resp = self.client.get(reverse('table_ram'), follow=True)
    self.assertEqual(resp.context['page_name'], "Оперативная память")
    self.assertEqual(resp.context['filter'], filters.TableRamFilter)

class ConfCmprViewTest(TestCase):

    def setUp(self):
        user = User.objects.create_user('temporary', 'temporary@gmail.com',
'temporary')
        models.ScrapperBenchRam.objects.create(
            name = "RamName",
            url = "someRAM.url",
            speed_read = 1598,
            speed_write = 8951,
            latency = 2587,
            in_stock = True,
            price = 999999
        )
        models.ScrapperBenchVideo.objects.create(
            name = "VgaName",
            url = "someVGA.url",
            score = 99999,
            rank = -1,
            in_stock = True,
            price = 999999
        )
        models.ScrapperBenchHDD.objects.create(
            name = "HddName",
            url = "someHDD.url",
            score = 99999,
            rank = -1,
            in_stock = True,
            price = 999999
        )
        models.ScrapperBenchCPU.objects.create(
            name = "CpuName",
            url = "someCPU.url",
            score = 99999,
            rank = -1,
            in_stock = True,
            price = 999999
        )
        models.ComputerConf.objects.create(
            cpu = models.ScrapperBenchCPU.objects.get(id=1),
            ram = models.ScrapperBenchRam.objects.get(id=1),
            hdd = models.ScrapperBenchHDD.objects.get(id=1),
            vga = models.ScrapperBenchVideo.objects.get(id=1)
        )

    def test_confcmpr_view_responce(self):
        self.client.login(username='temporary', password='temporary')

```

```

resp = self.client.get(reverse('conf_cmpr'), follow=True)
self.assertEqual(resp.status_code, 200)

def test_confcmpr_view_page_context(self):
    self.client.login(username='temporary', password='temporary')
    resp = self.client.get(reverse('conf_cmpr'), follow=True)
    self.assertEqual(
        range(0, models.ComputerConf.objects.all().count(), 1),
        resp.context['conf_count']
    )

class ConfCreateViewTest(TestCase):

    def setUp(self):
        user = User.objects.create_user('temporary', 'temporary@gmail.com',
'temporary')
        models.ScrapperBenchRam.objects.create(
            name = "RamName",
            url = "someRAM.url",
            speed_read = 1598,
            speed_write = 8951,
            latency = 2587,
            in_stock = True,
            price = 999999
        )
        models.ScrapperBenchVideo.objects.create(
            name = "VgaName",
            url = "someVGA.url",
            score = 99999,
            rank = -1,
            in_stock = True,
            price = 999999
        )
        models.ScrapperBenchHDD.objects.create(
            name = "HddName",
            url = "someHDD.url",
            score = 99999,
            rank = -1,
            in_stock = True,
            price = 999999
        )
        models.ScrapperBenchCPU.objects.create(
            name = "CpuName",
            url = "someCPU.url",
            score = 99999,
            rank = -1,
            in_stock = True,
            price = 999999
        )
        models.ComputerConf.objects.create(
            cpu = models.ScrapperBenchCPU.objects.get(id=1),
            ram = models.ScrapperBenchRam.objects.get(id=1),
            hdd = models.ScrapperBenchHDD.objects.get(id=1),
            vga = models.ScrapperBenchVideo.objects.get(id=1)
        )

    def test_confcmpr_view_page_form(self):
        self.client.login(username='temporary', password='temporary')
        resp = self.client.get(reverse('conf_create'), follow=True)
        test_date = models.ComputerConf(
            cpu = models.ScrapperBenchCPU.objects.all().first(),

```

```
        ram = models.ScrapperBenchRam.objects.all().first(),
        hdd = models.ScrapperBenchHDD.objects.all().first(),
        vga = models.ScrapperBenchVideo.objects.all().first(),
    )
    # test_form = forms.ConfCreateForm(instance=test_date)
    test_form = forms.ConfCreateForm()
    self.assertEqual(type(resp.context["form"]), type(test_form))
```

## Приложение В. Исходный код программы

В приложении приведен исходный код разработанной информационной системы. Поскольку код разбит по модулям и файлам – в приложении так же производится разбиение по модулям и файлам.

### В.1. Модуль view.py

```
from . import models
from . import filters
from django.shortcuts import render
from django.views.generic import TemplateView, DeleteView, UpdateView
from django.views.generic import FormView
from django.core.paginator import Paginator
from django_tables2 import MultiTableMixin, RequestConfig, SingleTableMixin,
SingleTableView
from django_tables2.views import SingleTableMixin
from django_filters.views import FilterView
from django.contrib import messages
from django.urls import reverse_lazy
from django.contrib.auth.mixins import LoginRequiredMixin
from .tables import CPUScoreTable, HDDTable, VGAScoreTable, RAMTable
from . import forms

class IndexPage(LoginRequiredMixin, TemplateView):
    template_name = "vergleich/index.html"
    flag_main = 'class=active'
    login_url = '/accounts/login/'
    redirect_field_name = 'redirect_to'

class TableCpu(LoginRequiredMixin, SingleTableMixin, FilterView):
    template_name = "vergleich/table_view.html"
    flag_table_cpu = 'class=active'
    model = models.ScraperBenchCPU
    queryset = models.ScraperBenchCPU.objects.all().order_by('-in_stock', '-score')
    table_class = CPUScoreTable
    paginate_by = 10
    filterset_class = filters.TableCPUFilter
    login_url = '/accounts/login/'
    redirect_field_name = 'redirect_to'

    def get_context_data(self, **kwargs):
        context = super(TableCpu, self).get_context_data(**kwargs)
        context['page_name'] = "Центральные процессоры"
        context['filter'] = self.filterset_class
        return context

class TableHDD(LoginRequiredMixin, SingleTableMixin, FilterView):
    template_name = "vergleich/table_view.html"
    flag_table_hdd = 'class=active'
    queryset = models.ScraperBenchHDD.objects.all().order_by('-in_stock', '-score')
    table_class = HDDTable
    paginate_by = 10
```



```

filterset_class = filters.TableHDDFilter
login_url = '/accounts/login/'
redirect_field_name = 'redirect_to'

def get_context_data(self, **kwargs):
    context = super(TableHDD, self).get_context_data(**kwargs)
    context['page_name'] = "Жёсткие диски"
    context['filter'] = self.filterset_class
    return context

class TableVGA(LoginRequiredMixin, SingleTableMixin, FilterView):
    template_name = "vergleich/table_view.html"
    flag_table_vga = 'class=active'
    model = models.ScrapperBenchVideo
    queryset = models.ScrapperBenchVideo.objects.all().order_by('-in_stock', '-score')
    table_class = VGATable
    paginate_by = 10
    filterset_class = filters.TableVGAFilter
    login_url = '/accounts/login/'
    redirect_field_name = 'redirect_to'

    def get_context_data(self, **kwargs):
        context = super(TableVGA, self).get_context_data(**kwargs)
        context['page_name'] = "Видеоускорители"
        context['filter'] = self.filterset_class
        return context

class TableRAM(LoginRequiredMixin, SingleTableMixin, FilterView):
    template_name = "vergleich/table_view.html"
    flag_table_ram = 'class=active'
    model = models.ScrapperBenchRam
    queryset = models.ScrapperBenchRam.objects.all().order_by('-in_stock', '-speed_read')
    table_class = RAMTable
    paginate_by = 15
    filterset_class = filters.TableRamFilter
    login_url = '/accounts/login/'
    redirect_field_name = 'redirect_to'
    ram_table = "True"

    def get_context_data(self, **kwargs):
        context = super(TableRAM, self).get_context_data(**kwargs)
        context['page_name'] = "Оперативная память"
        context['filter'] = self.filterset_class
        return context

class ConfCreateView(LoginRequiredMixin, FormView):
    template_name = "vergleich/create_view.html"
    form_class = forms.ConfCreateForm
    success_url = "."
    req = {}
    type_of_view = 'FormView'
    login_url = '/accounts/login/'
    redirect_field_name = 'redirect_to'

    def form_valid(self, form):
        self.req["form"] = form
        new_obj = models.ComputerConf(
            cpu = form.cleaned_data['cpu'],

```

```

        ram = form.cleaned_data['ram'],
        hdd = form.cleaned_data['hdd'],
        vga = form.cleaned_data['vga']
    )
    new_obj.save()
    return super().form_valid(form)

def get_context_data(self, **kwargs):
    ret = super(ConfCreateView, self).get_context_data(**kwargs)
    if self.req.get("form", None) != None:
        ret["form"] = self.req["form"]
        self.req["form"] = None
    return ret

class ConfCmprView(LoginRequiredMixin, TemplateView):
    template_name = "vergleich/compare_view.html"
    login_url = '/accounts/login/'
    redirect_field_name = 'redirect_to'

    def get_context_data(self, **kwargs):
        ret = super(ConfCmprView, self).get_context_data(**kwargs)
        ret['table_obj'] = models.ComputerConf.objects.all()
        ret['conf_count'] = range(0, models.ComputerConf.objects.all().count(), 1)
        return ret

class ConfUpdate(LoginRequiredMixin, UpdateView):
    model = models.ComputerConf
    template_name = "vergleich/create_view.html"
    fields = ['cpu', 'ram', 'vga', 'hdd']
    success_url = reverse_lazy('conf_cmpr')
    type_of_view = 'UpdateView'
    login_url = '/accounts/login/'
    redirect_field_name = 'redirect_to'

class ConfDelete(LoginRequiredMixin, DeleteView):
    model = models.ComputerConf
    success_url = reverse_lazy('conf_cmpr')
    login_url = '/accounts/login/'
    redirect_field_name = 'redirect_to'

```

## B.2. Модуль `vergleich\urls.py`

```

from django.conf.urls import url
from django.urls import register_converter, path, include

from . import views

urlpatterns = [
    url(r'^$', views.IndexPage.as_view(), name='index'),
    url(r'^accounts/', include('django.contrib.auth.urls')),

    # Tables
    url(r'^table_cpu/$', views.TableCpu.as_view(), name='table_cpu'),
    url(r'^table_hdd/$', views.TableHDD.as_view(), name='table_hdd'),
    url(r'^table_vga/$', views.TableVGA.as_view(), name='table_vga'),

```

```

url(r'^table_ram/$', views.TableRAM.as_view(), name='table_ram'),

# Compare
url(r'^create/$', views.ConfCreateView.as_view(), name='conf_create'),
url(r'^compare/$', views.ConfCmprView.as_view(), name='conf_cmpr'),
path('compare/<int:pk>/update/', views.ConfUpdate.as_view(),
name='conf_update'),
path('compare/<int:pk>/delete/', views.ConfDelete.as_view(),
name='conf_delete'),

# url(r'^book/(?P<pk>\d+)$', views.BookDetailView.as_view(), name='book-
detail'),
]

```

### B.3. Модуль tables.py

```

import django_tables2 as tables
from django.utils.html import format_html

from .models import ScrapperBenchCPU, ScrapperBenchHDD, ScrapperBenchRam,
ScrapperBenchVideo

class ColumnBool(tables.Column):
    def render(self, value):
        if value == "True" or value == True:
            return format_html('<i class="menu-icon fa fa-check-square-o">')
        else:
            return format_html('<i class="menu-icon fa fa-square-o">')

class ColumnNoPrice(tables.Column):
    def render(self, value):
        if value == -1:
            return format_html('<i class="menu-icon fa fa-minus">')
        else:
            return value

class ColumnLink(tables.Column):
    def render(self, value, record):
        return format_html('<a href="{}" target="_blank">{}</a>', record.url,
value)

class CPUScoreTable(tables.Table):
    name = ColumnLink(verbose_name="Имя")
    score = tables.Column(verbose_name="Индекс производительности")
    rank = tables.Column(verbose_name="Позиция")
    in_stock = ColumnBool(verbose_name="В наличии")
    price = ColumnNoPrice(verbose_name="Цена")

    class Meta:
        model = ScrapperBenchCPU
        fields = ('name', 'score', 'rank', 'in_stock', 'price')
        attrs = {'class': 'table table-striped'}

class HDDTable(tables.Table):
    name = ColumnLink(verbose_name="Имя")

```

```

score = tables.Column(verbose_name="Индекс производительности")
rank = tables.Column(verbose_name="Позиция")
in_stock = ColumnBool(verbose_name="В наличии")
price = ColumnNoPrice(verbose_name="Цена")

class Meta:
    model = ScrapperBenchHDD
    fields = ('name', 'score', 'rank', 'in_stock', 'price')
    attrs = {'class': 'table table-striped'}

class VGATable(tables.Table):
    name = ColumnLink(verbose_name="Имя")
    score = tables.Column(verbose_name="Индекс производительности")
    rank = tables.Column(verbose_name="Позиция")
    in_stock = ColumnBool(verbose_name="В наличии")
    price = ColumnNoPrice(verbose_name="Цена")

    class Meta:
        model = ScrapperBenchVideo
        fields = ('name', 'score', 'rank', 'in_stock', 'price')
        attrs = {'class': 'table table-striped'}

class RAMTable(tables.Table):
    name = ColumnLink(verbose_name="Имя")
    speed_read = tables.Column(verbose_name="Скорость чтения")
    speed_write = tables.Column(verbose_name="Скорость записи")
    latency = tables.Column(verbose_name="Задержка")
    in_stock = ColumnBool(verbose_name="В наличии")
    price = ColumnNoPrice(verbose_name="Цена")

    class Meta:
        model = ScrapperBenchVideo
        fields = ('name', 'speed_read', 'speed_write', 'latency', 'in_stock',
'price')
        attrs = {'class': 'table table-striped'}

```

## B.4. Модуль models.py

```

from django.core.exceptions import ValidationError
from django.utils.html import format_html
from django.db import models

# RAM table
class ScrapperBenchRam(models.Model):
    name = models.CharField(max_length=1000, blank=True)
    url = models.URLField(max_length=1000, blank=True)
    speed_read = models.IntegerField(blank=True)
    speed_write = models.IntegerField(blank=True)
    latency = models.IntegerField(blank=True)
    in_stock = models.BooleanField(default=False)
    price = models.FloatField(blank=True, default=-1)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['name']

```

```

# CPU table
class ScrapperBenchCPU(models.Model):
    name = models.CharField(max_length=1000, blank=True)
    url = models.URLField(max_length=1000, blank=True)
    score = models.FloatField(blank=True)
    rank = models.IntegerField(blank=True)
    in_stock = models.BooleanField(default=False)
    price = models.FloatField(blank=True)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['name']

# HDD table
class ScrapperBenchHDD(models.Model):
    name = models.CharField(max_length=1000, blank=True)
    url = models.URLField(max_length=1000, blank=True)
    score = models.FloatField(blank=True)
    rank = models.IntegerField(blank=True)
    in_stock = models.BooleanField(default=False)
    price = models.FloatField(blank=True)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['name']

# Video table
class ScrapperBenchVideo(models.Model):
    name = models.CharField(max_length=1000, blank=True)
    url = models.URLField(max_length=1000, blank=True)
    score = models.FloatField(blank=True)
    rank = models.IntegerField(blank=True)
    in_stock = models.BooleanField(default=False)
    price = models.FloatField(blank=True)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['name']

class ComputerConf(models.Model):
    cpu = models.ForeignKey('ScrapperBenchCPU', on_delete=models.CASCADE, blank =
False, null = False, verbose_name = "Цетральный процессор")
    ram = models.ForeignKey('ScrapperBenchRam', on_delete=models.CASCADE, blank =
False, null = False, verbose_name = "Оперативная память")
    hdd = models.ForeignKey('ScrapperBenchHDD', on_delete=models.CASCADE, blank =
False, null = False, verbose_name = "Жёсткий диск")
    vga = models.ForeignKey('ScrapperBenchVideo', on_delete=models.CASCADE, blank
= False, null = False, verbose_name = "Видеоускоритель")

    def __str__(self):
        return "{0}; {1}; {2}; {3}".format(self.cpu, self.ram, self.hdd, self.vga)

```

```

def clean(self):
    if ComputerConf.objects.all().count() > 6:
        # raise ValidationError('Too many records in model')
        raise ValidationError(format_html('<div class="alert alert-warning alert-dismissible fade show" role="alert">Вы можете добавлять не более 7 конфигураций.<button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-hidden="true">×</span></button></div>'))

```

## B.5. Модуль forms.py

```

from django import forms
from . import models

```

```

class ConfCreateForm(forms.ModelForm):

```

```

    class Meta:
        model = models.ComputerConf
        fields = '__all__'

```

## B.6. Модуль filters.py

```

from . import models
import django_filters

```

```

class DRYFilter(django_filters.FilterSet):
    name = django_filters.CharFilter(lookup_expr='icontains')
    score = django_filters.RangeFilter()
    rank = django_filters.NumberFilter()
    in_stock = django_filters.BooleanFilter()
    price = django_filters.RangeFilter()

```

```

class TableCPUFilter(DRYFilter):

```

```

    class Meta:
        model = models.ScrapperBenchCPU
        fields = ['name', 'score', 'rank', 'in_stock', 'price']

```

```

class TableHDDFilter(DRYFilter):

```

```

    class Meta:
        model = models.ScrapperBenchHDD
        fields = ['name', 'score', 'rank', 'in_stock', 'price']

```

```

class TableVGAFilter(DRYFilter):

```

```

    class Meta:
        model = models.ScrapperBenchVideo
        fields = ['name', 'score', 'rank', 'in_stock', 'price']

```

```

class TableRamFilter(django_filters.FilterSet):
    name = django_filters.CharFilter(lookup_expr='icontains')
    speed_read = django_filters.RangeFilter()

```

```

speed_write = django_filters.RangeFilter()
latency = django_filters.RangeFilter()
in_stock = django_filters.BooleanFilter()
price = django_filters.RangeFilter()

class Meta:
    model = models.ScrapperBenchRam
    fields = ['name', 'speed_read', 'speed_write', 'latency', 'in_stock',
'price']

```

## B.7. Файл base.html

```

<!doctype html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7" lang="">
<![endif]-->
<!--[if IE 7]>      <html class="no-js lt-ie9 lt-ie8" lang=""> <![endif]-->
<!--[if IE 8]>      <html class="no-js lt-ie9" lang=""> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js" lang=""> <!--<![endif]-->
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Vergleich</title>
    <meta name="description" content="Sufee Admin - HTML5 Admin Template">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    {% load static %}

    <link rel="apple-touch-icon" href="{% static 'apple-icon.png' %}">
    <link rel="shortcut icon" href="{% static 'favicon.ico' %}">

    <link rel="stylesheet" href="{% static 'assets/css/normalize.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/bootstrap.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/font-awesome.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/themify-icons.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/flag-icon.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/cs-skin-elastic.css' %}">
    {% block page-css %}{% endblock %}
    <!-- <link rel="stylesheet" href="assets/css/bootstrap-select.less"> -->
    <link rel="stylesheet" href="{% static 'assets/scss/style.css' %}">
    <link href="{% static 'assets/css/lib/vector-map/jqvm.min.css' %}"
rel="stylesheet">

    <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,600,700,800'
rel='stylesheet' type='text/css'>

    <!-- <script type="text/javascript"
src="https://cdn.jsdelivr.net/html5shiv/3.7.3/html5shiv.min.js"></script> -->

</head>
<body>

    <!-- Left Panel -->

    <aside id="left-panel" class="left-panel">
        <nav class="navbar navbar-expand-sm navbar-default">

            <div class="navbar-header">

```

```

        <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#main-menu" aria-controls="main-menu" aria-
expanded="false" aria-label="Toggle navigation">
        <i class="fa fa-bars"></i>
    </button>
    <!-- <a class="navbar-brand" href="./"></a> -->
    <a class="navbar-brand" href="./">Vergleich</a>
    <!-- <a class="navbar-brand hidden" href="./"></a> -->
    <a class="navbar-brand hidden" href="./">V</a>
</div>

<div id="main-menu" class="main-menu collapse navbar-collapse">
    <ul class="nav navbar-nav">
        <li {{ view.flag_main }}>
            <a href="/"> <i class="menu-icon fa fa-
home"></i>Главная</a>
        </li>
        <h3 class="menu-title">Конфигурации</h3><!-- /.menu-title -->
        <li>
            <a href="{% url 'conf_create' %}"> <i class="menu-icon fa
fa-check-square-o"></i>Создание</a>
        </li>
        <li>
            <a href="{% url 'conf_cmp'r' %}"> <i class="menu-icon fa
fa-dashboard"></i>Сравнение</a>
        </li>
        <h3 class="menu-title">Оборудование</h3><!-- /.menu-title -->
        <li {{ view.flag_table_cpu}}>
            <a href="{% url 'table_cpu' %}"> <i class="menu-icon fa
fa-table"></i>Цетральные процессоры</a>
        </li>
        <li {{ view.flag_table_ram}}>
            <a href="{% url 'table_ram' %}"> <i class="menu-icon fa
fa-table"></i>Оперативная память</a>
        </li>
        <li {{ view.flag_table_vga}}>
            <a href="{% url 'table_vga' %}"> <i class="menu-icon fa
fa-table"></i>Видеоускорители</a>
        </li>
        <li {{ view.flag_table_hdd}}>
            <a href="{% url 'table_hdd' %}"> <i class="menu-icon fa
fa-table"></i>Жёсткие диски</a>
        </li>
    </ul>
</div><!-- /.navbar-collapse -->
</nav>
</aside><!-- /#left-panel -->

<!-- Left Panel -->

<!-- Right Panel -->

<div id="right-panel" class="right-panel">

    <!-- Header-->
    <header id="header" class="header">

        <div class="header-menu">

            <div class="col-sm-7">

```



```

        <a id="menuToggle" class="menutoggle pull-left"><i class="fa
fa fa-tasks"></i></a>
        <div class="header-left">
        </div>
    </div>

    <div class="col-sm-5">
        <div class="user-area dropdown float-right">
            <a href="#" class="dropdown-toggle" data-toggle="dropdown"
aria-haspopup="true" aria-expanded="false">
                
            </a>

            <div class="user-menu dropdown-menu">
                <a class="nav-link" href="{% url 'logout'
%}?next={{request.path}}"><i class="fa fa-power-off"></i> Выйти</a>
            </div>
        </div>

        <div class="language-select dropdown" id="language-select">
            <a class="dropdown-toggle" href="#" data-toggle="dropdown"
id="language" aria-haspopup="true" aria-expanded="true">
                <i class="flag-icon flag-icon-ru"></i>
            </a>
        </div>

    </div>
</div>

</header><!-- /header -->
<!-- Header-->

<div class="breadcrumbs">
    <div class="col-sm-4">
        <div class="page-header float-left">
            <div class="page-title">
                <h1>
                    {% block page-title %}Заголовок страницы{% endblock %}
                </h1>
            </div>
        </div>
    </div>
    <div class="col-sm-8">
        <div class="page-header float-right">
            <div class="page-title">
                <ol class="breadcrumb text-right">
                    <li class="active">{{ user.get_username }}</li>
                </ol>
            </div>
        </div>
    </div>
</div>

<div class="content mt-3">

    {% block content %}
    {% endblock %}

</div> <!-- .content -->
</div><!-- /#right-panel -->

<!-- Right Panel -->

```

```

        <script src="{% static 'assets/js/vendor/jquery-2.1.4.min.js' %}"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper.min.js"></
script>
        <script src="{% static 'assets/js/plugins.js' %}"></script>
        <script src="{% static 'assets/js/main.js' %}"></script>

        <script src="{% static 'assets/js/lib/chart-js/Chart.bundle.js' %}"></script>
        <script src="{% static 'assets/js/dashboard.js' %}"></script>
        <script src="{% static 'assets/js/widgets.js' %}"></script>
        <script src="{% static 'assets/js/lib/vector-map/jquery.vmap.js' %}"></script>
        <script src="{% static 'assets/js/lib/vector-map/jquery.vmap.min.js'
%}"></script>
        <script src="{% static 'assets/js/lib/vector-map/jquery.vmap.sampledata.js'
%}"></script>
        <script src="{% static 'assets/js/lib/vector-map/country/jquery.vmap.world.js'
%}"></script>
        <script>
            ( function ( $ ) {
                "use strict";

                jQuery( '#vmap' ).vectorMap( {
                    map: 'world_en',
                    backgroundColor: null,
                    color: '#ffffff',
                    hoverOpacity: 0.7,
                    selectedColor: '#1de9b6',
                    enableZoom: true,
                    showTooltip: true,
                    values: sample_data,
                    scaleColors: [ '#1de9b6', '#03a9f5' ],
                    normalizeFunction: 'polynomial'
                } );
            } )( jQuery );
        </script>
        {% block js %}
        {% endblock %}
    </body>
</html>

```

## B.8. Файл compare\_view.html

```

{% extends "vergleich/base.html" %}
{% load staticfiles %}

{% block page-title %}Сравнение конфигураций{% endblock %}

{% block content %}

<div class="col-md-12">
    <div class="feed-box text-center">
        <section class="card">
            <div class="card-body">
                <table class="table">
                    <tr>
                        <th></th>
                        {% for i in conf_count %}

```

```

        <th>Конф. {{ forloop.counter }}</th>
        {% endfor %}
    </tr>
    <tr>
        <td>Модель процессора</td>
        {% for data in table_obj %}
            <td>{{ data.cpu.name }} <a href="{{ data.cpu.url }}"
target="_blank"><i class="menu-icon fa fa-link"></i></a></td>
            {% endfor %}
        </tr>
        <tr>
            <td>Индекс производительности</td>
            {% for data in table_obj %}
                <td>{{ data.cpu.score }}</td>
            {% endfor %}
        </tr>
        <tr>
            <td>Модель оперативной памяти</td>
            {% for data in table_obj %}
                <td>{{ data.ram.name }} <a href="{{ data.ram.url }}"
target="_blank"><i class="menu-icon fa fa-link"></i></a></td>
            {% endfor %}
        </tr>
        <tr>
            <td>Скорость чтения</td>
            {% for data in table_obj %}
                <td>{{ data.ram.speed_read }}</td>
            {% endfor %}
        </tr>
        <tr>
            <td>Скорость записи</td>
            {% for data in table_obj %}
                <td>{{ data.ram.speed_write }}</td>
            {% endfor %}
        </tr>
        <tr>
            <td>Задержка</td>
            {% for data in table_obj %}
                <td>{{ data.ram.latency }}</td>
            {% endfor %}
        </tr>
        <tr>
            <td>Модель видеоускорителя</td>
            {% for data in table_obj %}
                <td>{{ data.vga.name }} <a href="{{ data.vga.url }}"
target="_blank"><i class="menu-icon fa fa-link"></i></a></td>
            {% endfor %}
        </tr>
        <tr>
            <td>Индекс производительности</td>
            {% for data in table_obj %}
                <td>{{ data.vga.score }}</td>
            {% endfor %}
        </tr>
        <tr>
            <td>Модель жёсткого диска</td>
            {% for data in table_obj %}
                <td>{{ data.hdd.name }} <a href="{{ data.hdd.url }}"
target="_blank"><i class="menu-icon fa fa-link"></i></a></td>
            {% endfor %}
        </tr>
        <tr>
            <td>Индекс производительности</td>

```

```

        {% for data in table_obj %}
            <td>{{ data.hdd.score }}</td>
        {% endfor %}
    </tr>
    <tr>
        <td></td>
        {% for data in table_obj %}
            <td>
                <a href="{{ data.pk }}/update/"><i class="fa fa-
pencil"></i></a>
                <a href="{{ data.pk }}/delete/"><i class="fa fa-
times-circle"></i></a>
            </td>
        {% endfor %}
    </tr>
</table>
</div>
</section>
</div>
</div>
{% endblock %}

```

## В.9. Файл computerconf\_confirm\_delete.html

```

{% extends "vergleich/base.html" %}
{% load staticfiles %}

{% block page-title %}Удаление записи{% endblock %}

{% block content %}

<div class="col-md-4">
    <div class="card">
        <div class="card-header">
            <strong class="card-title">Подтвердите удаление</strong>
        </div>
        <form action="" method="POST">
            <div class="card-body">
                <p class="card-text">
                    <p>Вы действительно хотите удалить конфигурацию?</p>
                </div>
                <div class="card-footer">
                    {% csrf_token %}
                    <table style="width: 100%;">
                        <tr>
                            <td style="width: 47%;">
                                <input type="submit" action="" value="Да, удалить!"
class="btn btn-outline-danger btn-lg btn-block"/>
                            </td>
                            <td style="width: 6%;"></td>
                            <td style="width: 47%;">
                                <a href="{% url 'conf_cmpr' %}"><button type="button"
class="btn btn-outline-secondary btn-lg btn-block">Нет</button></a>
                            </td>
                        </tr>
                    </table>
                </div>
            </form>

```

```

    </div>
</div>

{% endblock %}

```

## B.10. Файл create\_view.html

```

{% extends "vergleich/base.html" %}
{% load staticfiles %}
{% load django_tables2 %}

{% block page-css %}
<link rel="stylesheet" href="{% static 'assets/css/vergleich_style.css' %}">

<style>
    div.col-md-9{
        display: none;
    }
</style>

{% endblock %}

{% block page-title %}Создание конфигурации{% endblock %}

{% block content %}

<div class="col-md-9">
    <div class="card">
        <div class="card-header">
            <strong class="card-title">
                {% if view.type_of_view == 'FormView' %}
                    Заполните форму для добавления конфигурации в таблицу
сравнения
                {% elif view.type_of_view == 'UpdateView' %}
                    Измените компоненты и сохраните конфигурацию
                {% endif %}
            </strong>
        </div>
        <form action="" method="post">
            <div class="card-body">
                <p class="card-text">
                    {% csrf_token %}
                    <table>
                        {{ form.as_table }}
                    </table>
                </p>
            </div>
            <div class="card-footer">

                {% if view.type_of_view == 'FormView' %}
                    <input type="submit" class="btn btn-outline-primary btn-lg
btn-block"></input>
                {% elif view.type_of_view == 'UpdateView' %}
                    <input type="submit" class="btn btn-outline-success btn-lg
btn-block"></input>
                {% endif %}

            </div>
        </form>
    </div>
</div>

```

```

        </form>
    </div>
</div>

{% endblock %}

{% block js %}

    <script>
        jQuery(document).ready(function($) {
            $("#id_cpu").addClass("form-control-sm form-control")
            $("#id_ram").addClass("form-control-sm form-control")
            $("#id_hdd").addClass("form-control-sm form-control")
            $("#id_vga").addClass("form-control-sm form-control")
            $(".table").addClass("table")
            $(".col-md-9").css("display", "initial")
        });
    </script>

{% endblock %}

```

## B.11. Файл index.html

```

{% extends "vergleich/base.html" %}
{% load staticfiles %}

{% block page-title %}Главная{% endblock %}

{% block content %}

<div class="col-md-12">
    <div class="card">
        <div class="card-header">
            <strong class="card-title">Приветствую!</strong>
        </div>
        <div class="card-body">
            <p class="card-text">
                <div class="col-md-9">
                    <p>
                        Вы находитесь на главной странице программы сравнения
конфигураций компьютеров.
                    </p>
                    <p>
                        Современное общество невозможно представить без
компьютеров. Будь то обычная сим-карта или дата-центр в несколько десятков гектар
площадью,
                        смартфоны, настольные решения, планшеты, игровые автоматы,
сложные системы управления технологическими линиями – всё это представляет собой
компьютер в том или ином виде. Компьютеризация несомненно
затронула все сферы жизнедеятельности человечества. Вычислительные машины являются
мощным инструментом, который упрощает нашу жизнь.
Компьютеру не нужен отдых, а вышедшие из строя детали легко заменить. А в
последние годы
                        складывается тенденция к развитию слабой форме
искусственного интеллекта – когда компьютер начинает делать выводы по решаемой
задаче самостоятельно,
                        что уже применяется в области медицины и астрономии.
                    </p>
                </p>
            </div>
        </div>
    </div>

```

Среди массового потребителя очень популярно решение в виде стационарного домашнего компьютера, который позволяет выполнять игровые и мультимедийные функции. Чаще всего представляет собой совокупность нескольких компонентов: системный блок, монитор, манипуляторы ввода (клавиатура, мышь), колонки. Подобная конфигурация позволяет легко заменить любой из компонентов самостоятельно, не обладая специфичными

знаниями или инструментом. Компоненты системного блока так же представляют из себя отдельные аппаратные решения, которые можно заменить.

Если выбор периферийных устройств не представляет сложную задачу, то выбор компонентов системного блока – задача требующего особых знаний.

Данная программа позволяет формировать конфигурации системного блока с учётом производительности компонентов. На соответствующих страницах

Вы можете как **[создать]({% url 'conf_create' %})** конфигурацию, так и **[сравнить]({% url 'conf_cmpr' %})** сохранённые ранее конфигурации.

```

    </p>
  </div>
  <div class="col-md-3">
    <a href="https://www.modders-inc.com/green-way-hostiti-
modding/" target="_blank"></a>
  </div>
</p>
</div>
</div>
</div>
{% endblock %}

```

## B.12. Файл table\_view.html

```

{% extends "vergleich/base.html" %}
{% load staticfiles %}
{% load django_tables2 %}

{% block page-css %}
<style>
  div.col-md-3{
    display: none;
  }
</style>
{% endblock %}

{% block page-title %}Сводная таблица{% endblock %}

{% block content %}

<div class="col-md-9">
  <div class="feed-box text-center">
    <section class="card">
      <div class="card-body">
        <h2>{{ page_name }}</h2>
        <p>
          {% render_table table %}
        </p>
      </div>
    </section>
  </div>
</div>
</div>

```

```

<div class="col-md-3">
  <div class="card">
    <div class="card-header">
      <strong class="card-title">Фильтр</strong>
    </div>
    <div class="card-body">
      <!-- <p class="card-text"> -->
        <form action="" method="get">
          {{ filter.form.as_p }}
          <input type="submit" class="btn btn-outline-success btn-
sm"></input>
        </form>
      <!-- </p> -->
    </div>
  </div>
</div>

{% endblock %}

{% block js %}

<script>
  jQuery(document).ready(function($) {
    $("#id_name").addClass("form-control col-sm-12");
    $("#id_name").prev().text("Имя содержит");

    {% if view.ram_table == "True" %}
      $("#id_speed_read_0").addClass("form-control");
      $("#id_speed_read_1").addClass("form-control");
      $("#id_speed_read_0").prev().text("Скорость чтения между");
      $("#id_speed_read_0").attr("placeholder", "Начиная с ...");
      $("#id_speed_read_1").attr("placeholder", "... до");
      $("#id_speed_write_0").addClass("form-control");
      $("#id_speed_write_1").addClass("form-control");
      $("#id_speed_write_0").prev().text("Скорость записи между");
      $("#id_speed_write_0").attr("placeholder", "Начиная с ...");
      $("#id_speed_write_1").attr("placeholder", "... до");
      $("#id_latency_0").addClass("form-control");
      $("#id_latency_1").addClass("form-control");
      $("#id_latency_contains").prev().text("Задержка между");
      $("#id_latency_0").attr("placeholder", "Начиная с ...");
      $("#id_latency_1").attr("placeholder", "... до");
    {% else %}
      $("#id_score_0").addClass("form-control");
      $("#id_score_0").attr("placeholder", "Начиная с ...");
      $("#id_score_0").prev().text("Индекс производительности между");
      $("#id_score_1").addClass("form-control");
      $("#id_score_1").attr("placeholder", "... до");
      $("#id_rank").addClass("form-control");
      $("#id_rank").prev().text("Позиция");
    {% endif %}

    $("#id_price_0").addClass("form-control");
    $("#id_price_0").attr("placeholder", "Начиная с ...");
    $("#id_price_0").prev().text("Цена между");
    $("#id_price_1").addClass("form-control");
    $("#id_price_1").attr("placeholder", "... до");
    $("#id_in_stock").addClass("form-control");
    $("#id_in_stock").prev().text("Наличие");

    $(".col-md-3").css("display", "initial");
  });

```



```
    });  
</script>  
  
{% endblock %}
```

## B.13. Модуль `urls.py`

```
from django.contrib import admin  
from django.urls import path  
from django.conf.urls import include  
from django.conf.urls import url  
from django.conf import settings  
from django.conf.urls.static import static  
from django.views.generic import RedirectView  
  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    url(r'^vergleich/', include('vergleich.urls')),  
    # Redirect to main page  
    url(r'^$', RedirectView.as_view(url='/vergleich/', permanent=True)),  
    # Login pages  
    url(r'^accounts/', include('django.contrib.auth.urls')),  
]  
# Static files  
urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```