



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ

Робототехника и комплексная автоматизация

КАФЕДРА

Системы автоматизированного проектирования (РК-6)

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***  
***НА ТЕМУ:***  
***«Разработка библиотеки функций, реализующей  
автоматизированное построение динамических  
графических пользовательских интерфейсов»***

Студент РК6-83Б  
(Группа)

\_\_\_\_\_  
(Подпись, дата) **А.Р. Василян**  
(И.О.Фамилия)

Руководитель ВКР

\_\_\_\_\_  
(Подпись, дата) **А.П. Соколов**  
(И.О.Фамилия)

Консультант

\_\_\_\_\_  
(Подпись, дата) (И.О.Фамилия)

Консультант

\_\_\_\_\_  
(Подпись, дата) (И.О.Фамилия)

Нормоконтролер

\_\_\_\_\_  
(Подпись, дата) (И.О.Фамилия)

2023 г.

**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

---

УТВЕРЖДАЮ  
Заведующий кафедрой РК-6  
(индекс)

\_\_\_\_\_  
А.П. Карпенко  
(инициалы и фамилия)

«\_\_» \_\_\_\_\_ 2023 г.

## **ЗАДАНИЕ** **на выполнение выпускной квалификационной работы бакалавра**

Студент группы: РК6-83Б

\_\_\_\_\_  
Василян Артур Размирович  
(фамилия, имя, отчество)

Тема квалификационной работы Разработка библиотеки функций, реализующей автоматизированное построение динамических графических пользовательских интерфейсов

---

---

Источник тематики (НИР кафедры, заказ организаций и т.п.): Кафедра

Тема квалификационной работы утверждена распоряжением по факультету РК № 6 от «8» октября 2023 г

***Часть 1. Аналитический обзор литературы.***

Должен быть выполнен аналитический обзор литературы, в рамках которого должны быть изучены существующие подходы разработки графических пользовательских интерфейсов. Должен быть сформулирован базовый принцип генерации графического пользовательского интерфейса и выбран формат входных данных.

***Часть 2. Программная реализация.***

Разработать программное решение для генерации графического пользовательского интерфейса. С использованием CMS Django разработать web-приложение (развернуть его с использованием docker).

***Часть 3. Тестирование.***

Используя сгенерированный интерфейс в web-приложении, запустить его на тестовом сервере `sandbox.rk6.bmstu.ru` и проверить корректность работы всего программного решения.

**Оформление квалификационной работы:**

Расчетно-пояснительная записка на 67 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

<i>количество: 15 рис., 36 листингов, 9 источн.</i>
---

Дата выдачи задания «08» октября 2023 г.

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до « \_\_\_\_ » \_\_\_\_\_ 20\_\_\_\_ г.

**Руководитель квалификационной работы**

_____	_____ <u>А.П. Соколова</u>
(Подпись, дата)	(И.О.Фамилия)

**Студент**

_____	_____ <u>А.Р. Василян</u>
(Подпись, дата)	(И.О.Фамилия)

Примечание:

1. Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Министерство науки и высшего образования Российской Федерации**  
**Федеральное государственное бюджетное образовательное учреждение**  
**высшего образования**  
**«Московский государственный технический университет имени Н.Э. Баумана**  
**(национальный исследовательский университет)»**  
**(МГТУ им. Н.Э. Баумана)**

**ФАКУЛЬТЕТ** РК  
**КАФЕДРА** РК-6  
**ГРУППА** РК6-83Б

**УТВЕРЖДАЮ**  
 Заведующий кафедрой РК-6  
 (индекс)

\_\_\_\_\_  
*А.П. Карпенко*  
 (инициалы и фамилия)

« \_\_\_\_ » \_\_\_\_\_ 2023 г.

**КАЛЕНДАРНЫЙ ПЛАН**  
**выполнения выпускной квалификационной работы**

студента: Василян Артур Размирович  
 (Фамилия, имя, отчество)

Тема квалификационной работы: Разработка библиотеки функций, реализующей автоматизированное построение динамических графических пользовательских интерфейсов

№ п/п	Наименование этапов выпускной квалификационной работы	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Должность	ФИО, подпись
1.	Задание на выполнение работы. Формулирование проблемы, цели и задач работы	<u>30.09.2022</u> Планируемая дата		Руководитель ВКР	<i>А.П. Соколов</i>
2.	1 часть <u>аналитический обзор литературы</u>	<u>30.09.2022</u> Планируемая дата	<u>15.10.2022</u>	Руководитель ВКР	<i>А.П. Соколов</i>
3.	Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач	<u>21.10.2022</u> Планируемая дата	<u>21.10.2022</u>	Заведующий кафедрой	<i>А.П. Соколов</i>
4.	2 часть <u>Программная реализация</u>	<u>25.11.2022</u> Планируемая дата	<u>04.05.2023</u>	Руководитель ВКР	<i>А.П. Соколов</i>
5.	3 часть <u>Тестирование</u>	<u>23.12.2022</u> Планируемая дата	<u>05.05.2023</u>	Руководитель ВКР	<i>А.П. Соколов</i>
6.	1-я редакция работы	<u>24.05.2023</u> Планируемая дата	<u>24.05.2023</u>	Руководитель ВКР	<i>А.П. Соколов</i>
7.	Подготовка доклада и презентации	_____ Планируемая дата			
8.	Заключение руководителя	_____ Планируемая дата		Руководитель ВКР	<i>А.П. Соколов</i>
9.	Нормоконтроль	_____ Планируемая дата		Нормоконтролер	<i>С.В. Грошев</i>
10.	Внешняя рецензия	_____ Планируемая дата			
11.	Защита работы на ГЭК	_____ Планируемая дата			

Студент \_\_\_\_\_  
 (подпись, дата)

Руководитель работы \_\_\_\_\_  
 (подпись, дата)

**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

---

**НАПРАВЛЕНИЕ  
НА ГОСУДАРСТВЕННУЮ ИТОГОВУЮ АТТЕСТАЦИЮ**

**Председателю  
Государственной Экзаменационной Комиссии №\_\_\_\_\_**

факультета «Робототехника и комплексная автоматизация» МГТУ им. Н.Э.Баумана

Направляется студент Василян Артур Размирович группы РК6-83Б

на защиту выпускной квалификационной работы «Разработка библиотеки функций, реализующей автоматизированное построение динамических графических пользовательских интерфейсов».

Декан факультета \_\_\_\_\_ «\_\_\_\_» \_\_\_\_\_ 20\_\_ г.

**Справка об успеваемости**

Студент Василян Артур Размирович за время пребывания в МГТУ имени Н.Э.Баумана с 2019 г. по 2023 г. полностью выполнил учебный план со следующими оценками: отлично – \_\_\_\_\_ %, хорошо – \_\_\_\_\_ %, удовлетворительно – \_\_\_\_\_ %.

Инспектор деканата \_\_\_\_\_

**Отзыв руководителя выпускной квалификационной работы**

Студент \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Руководитель ВКР \_\_\_\_\_ «\_\_\_\_» \_\_\_\_\_ 20\_\_ г.

(ФИО студента)

(подпись)

(дата)

## РЕФЕРАТ

выпускная квалификационная работа: 67 с., 15 рис., 36 листингов, 9 источн.

ГРАФИЧЕСКИЙ ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС, GUI, АВТОМАТИЗИРОВАННОЕ ПОСТРОЕНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА, ГЕНЕРАЦИЯ ИНТЕРФЕЙСА, DJANGO, DOCKER, PYPARSING, HTML, CSS.

Работа посвящена изучению методов построения графического пользовательского интерфейса и разработке библиотеки функций, реализующей автоматизированное построение графических пользовательских интерфейсов.

**Тип работы:** выпускная квалификационная работа.

**Тема работы:** *«Разработка библиотеки функций, реализующей автоматизированное построение динамических графических пользовательских интерфейсов».*

**Объект исследования:** Графический пользовательский интерфейс.

**Основная задача, на решение которой направлена работа:** Разработка библиотеки функций, реализующей автоматизированное построение динамических графических пользовательских интерфейсов.

**Цели работы:** рассмотреть существующие подходы к разработке графического пользовательского интерфейса, разработать библиотеку, обеспечивающую автоматизацию построения динамических пользовательских интерфейсов.

В результате выполнения работы: 1) были рассмотрены существующие подходы разработки GUI; 2) была разработана программа для преобразования данных формата aINI в HTML-код; 3) в рамках Django и Docker было разработано web-приложение. 4) Используя сгенерированный интерфейс в web-приложении, приложение было запущено на тестовом сервере `sandbox.rk6.bmstu.ru` и проверено корректность работы всего программного решения.

## **СОКРАЩЕНИЯ**

GUI – графический пользовательский интерфейс

DSL – предметно-ориентированный язык

URL – унифицированный указатель ресурса

CSS – каскадные таблицы стилей

JS – JavaScript

## СОДЕРЖАНИЕ

РЕФЕРАТ .....	6
ВВЕДЕНИЕ .....	9
1 Подходы к разработке пользовательского интерфейса.....	11
1.1 Метод построения оконного интерфейса пользователя на основе моделирования пользовательских целей .....	11
1.2 Методический подход к созданию средства построения пользовательского интерфейса .....	15
1.3 Построение пользовательского интерфейса с использованием интерактивного машинного обучения.....	18
2 Разработка web-приложения .....	19
2.1 Используемые программные решения.....	19
2.2 Разработка тестового web-приложения .....	21
2.3 Запуск web-приложения на тестовом сервере .....	28
3 Генерация GUI на основе aINI .....	31
3.1 Соответствие aINI кода с элементами интерфейса .....	31
3.2 Разработка программы .....	35
4 Тестирование .....	50
ЗАКЛЮЧЕНИЕ .....	66
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	67



## ВВЕДЕНИЕ

В процессе решения некоторой задачи необходимым оказывается создать форму ввода данных (пользовательский интерфейс), которые будут использоваться для решения этой задачи. Разработка такой формы ввода может быть не такой трудоемкой и время затратной, если требуется немного таких форм ввода и, если человек обладает соответствующими навыками программирования. Но если требуется большое количество таких форм ввода, которые в придачу имеют переключаемые вкладки, то время на разработку интерфейса значительно увеличится и усложнится, тем более для человека без навыков программирования.

В таких случаях разумным решением будет разработка инструмента для генерации таких форм ввода или интерфейса на основе простого описания элементов каждой формы.

Интерфейс [2] — это совокупность средств, методов и правил взаимодействия, управления, контроля и т.д. между элементами системы; Пользовательский интерфейс [2] — это разновидность интерфейсов, в котором одна сторона представлена человеком-пользователем, другая — машиной-устройством.

Графический пользовательский интерфейс [2] (GUI) — это разновидность пользовательского интерфейса, в котором элементы интерфейса, представленные пользователю на дисплее, исполнены в виде графических изображений.

На рис. 1 представлена схема базового принципа генерации GUI на основе данных на предметно-ориентированных языках (DSL). С помощью представленных на рисунке DSL (слева на схеме) можно описать элементы интерфейса для дальнейшей генерации на их основе. Наиболее простым и предпочтительным для обычного пользователя будет aINI [1] (формат для описания исходных данных различных задач, основанный на языке INI). Так как у него простой синтаксис, генерировать интерфейс сможет человек без знаний программирования.

## Возможные типы графических форм ввода

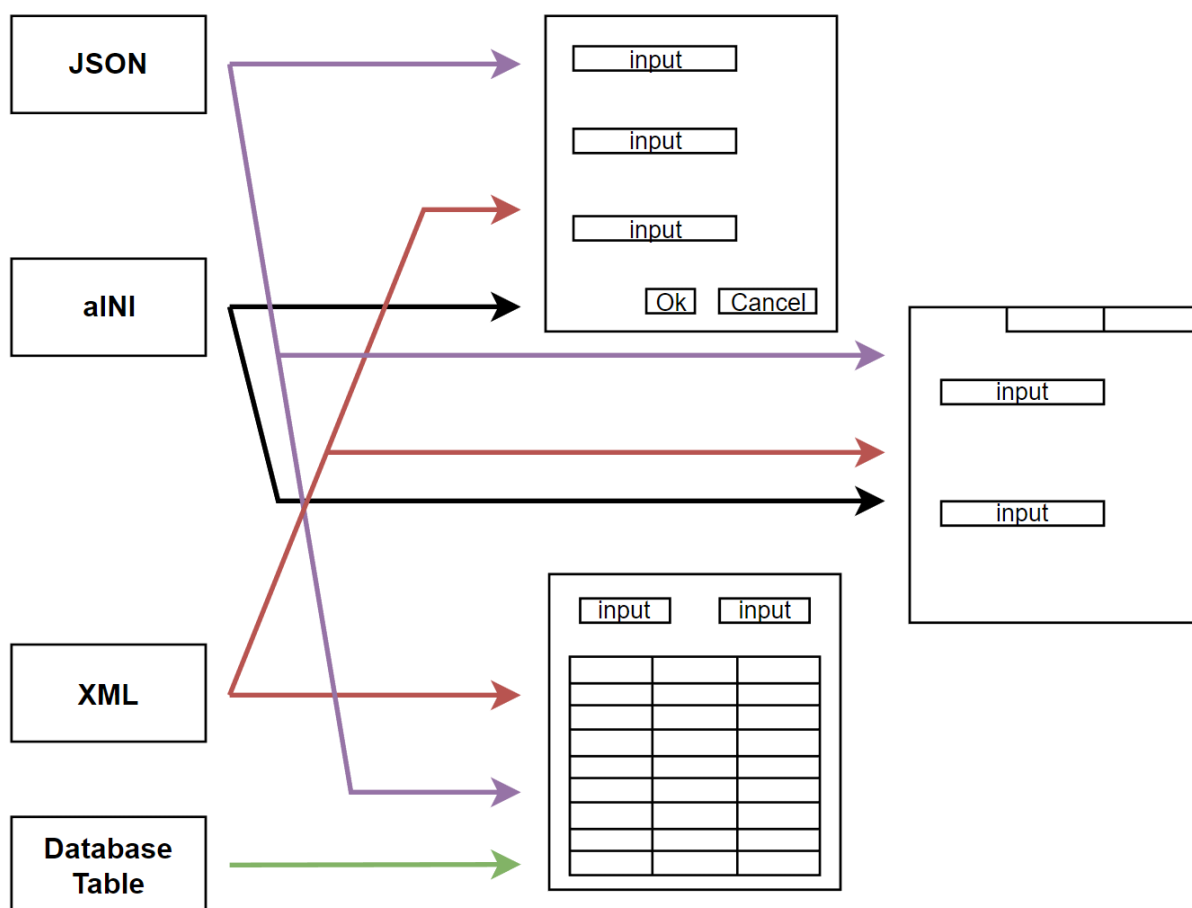


Рис. 1. Схема базового принципа генерации GUI на основе DSL

Следовательно, требуется разработать программу, которая принимает на вход данные в формате aINI, преобразует полученные данные и выводит в HTML-файл, который далее будет использован для разметки web-приложения.

## **1 Подходы к разработке пользовательского интерфейса**

### **1.1 Метод построения оконного интерфейса пользователя на основе моделирования пользовательских целей**

При изучении статьи [3] были выделены два метода взаимодействия пользователя и ЭВМ: ограничительный и направляющий.

Ограничительный метод.

Для средств взаимодействия пользователя и ЭВМ с оконным интерфейсом типичным является ограничительный метод. Метод заключается в том, что пользователю предоставляется некий набор операций, благодаря которым он может выполнять определенные действия с описанными в ЭВМ объектами своей деятельности. Операция имеет название, исходные данные и результаты, представляющие собой объекты воздействия операции. Пользователь решает, какую из операций необходимо выбрать в данный момент для выполнения своего задания, выбирает нужную операцию и задает для нее исходные данные. После чего ЭВМ выполняет указанную операцию, активируя соответствующие функции приложения, и выдаёт результаты операции пользователю.

По итогам выполнения очередной операции пользователь решает, какую следующую операцию ему нужно выбрать, передаёт ее ЭВМ на выполнение и т.д. Этот процесс он должен продолжать до тех пор, пока в итоге выполнения операций не будет достигнут желаемый результат, соответствующий выполненному заданию. Следовательно, пользователь должен сам планировать ход выполнения своего задания из предоставляемых ему операций. Обобщенная схема ограничительного метода взаимодействия приведена на рис. 2. Для ограничительного метода считается, что у пользователя присутствуют процедурные знания, необходимые для планирования процесса выполнения своих заданий.



пользовательского задания, совершает не пользователь, а ЭВМ. От пользователя требуется в ответ на запросы ЭВМ ввести исходные данные, необходимые для этих операций.

Направляющий метод взаимодействия “пользователь-ЭВМ” состоит из следующих основных этапов:

- информирование пользователя о множестве допустимых заданий, которые может выполнять ЭВМ в рамках данного приложения;
- выбор пользователем задания по меню заданий и передача его ЭВМ на выполнение;
- планирование процесса взаимодействия при выполнении задания;
- ввод пользователем данных, необходимых ЭВМ для выполнения задания;
- передача пользователю результатов выполнения задания и их оценка пользователем.

Обобщенная схема направляющего метода взаимодействия приведена на рис. 3. В соответствии с этой схемой пользователь должен выбрать в меню заданий свое задание и передать его ЭВМ на выполнение. Выбранное задание рассматривается в ЭВМ как цель пользователя. Достижение этой цели обеспечивается в ходе последующего диалога с пользователем, в котором инициатива взаимодействия принадлежит ЭВМ.

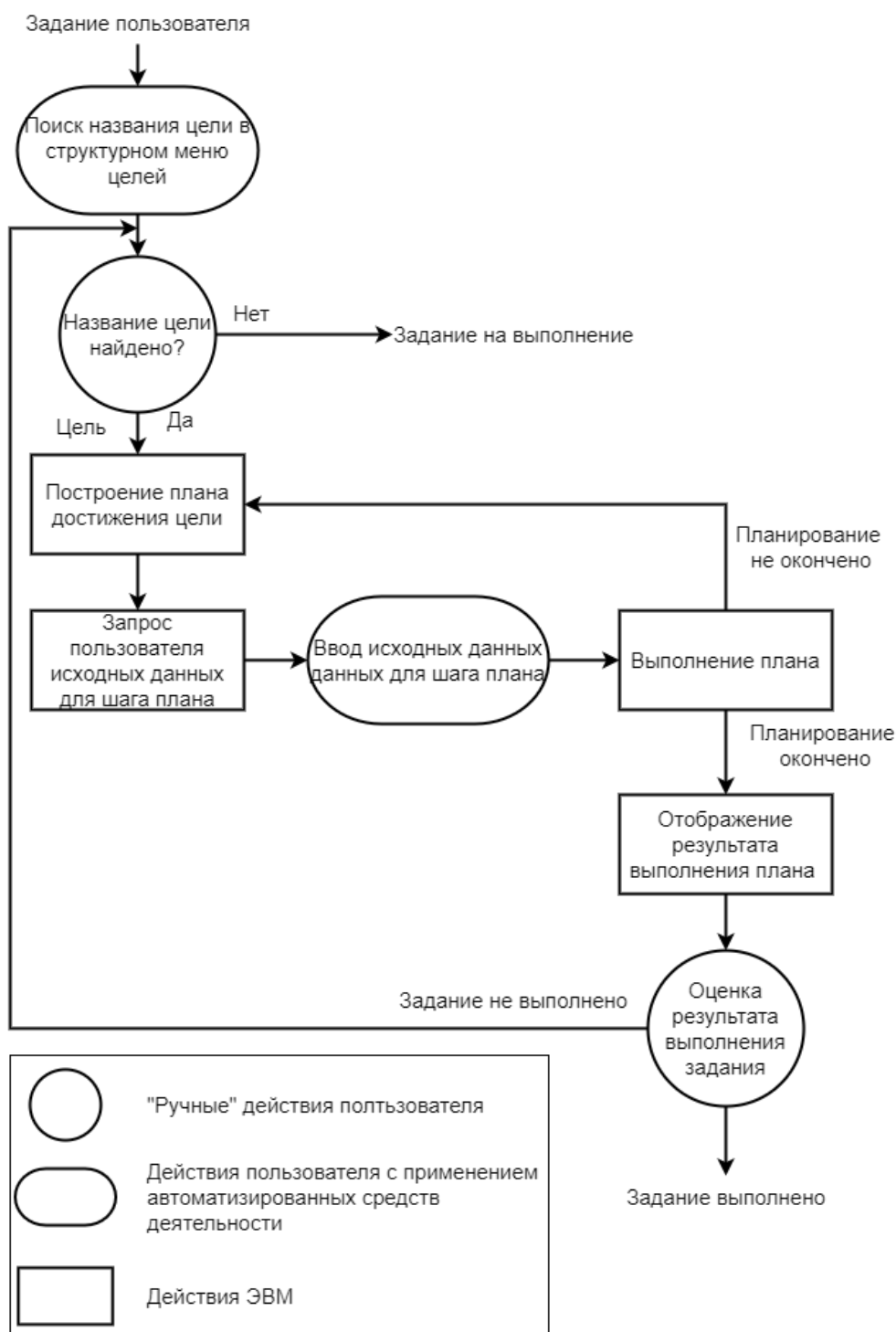


Рис. 3. Обобщённая схема направляющего метода взаимодействия “пользователь-ЭВМ” для оконного интерфейса

В отличие от схемы ограничительного метода, в рассматриваемой схеме существует только один маршрут движения, т. е. здесь нет зависимости результата от степени пользовательской подготовки. После передачи пользовательской цели в ЭВМ дальнейшее взаимодействие не требует инициативы пользователя вплоть до этапа оценки результата достижения цели. Планирование процесса взаимодействия на множестве ДТ осуществляет ЭВМ в соответствии с ДТ-моделью диалога. Данные, необходимые ЭВМ для

достижения принятой от пользователя цели, вводятся пользователем только по запросу, т.е. на этой стадии он выступает в роли реагирующего участника.

Итак, главное отличие направляющего метода взаимодействия “пользователь-ЭВМ” по сравнению с ограничительным методом состоит в том, что инициатива взаимодействия по достижению цели пользователя принадлежит не пользователю, а ЭВМ. Это означает, что ЭВМ играет активную роль в целенаправленном взаимодействии по выполнению задания пользователя.

Сопоставление схемы направляющего метода взаимодействия (рис. 2) со схемой ограничительного метода (рис. 3) позволяет наглядно судить об упрощении работы пользователя за счет того, что пользователю уже не требуется знать, как выполнить то или иное задание, поскольку планирование процесса взаимодействия выполняет не он сам, а ЭВМ.

## 1.2 Методический подход к созданию средства построения пользовательского интерфейса

При изучении статьи [4] были выделены составные элементы подхода к созданию универсального средства построения пользовательского интерфейса программных средств.

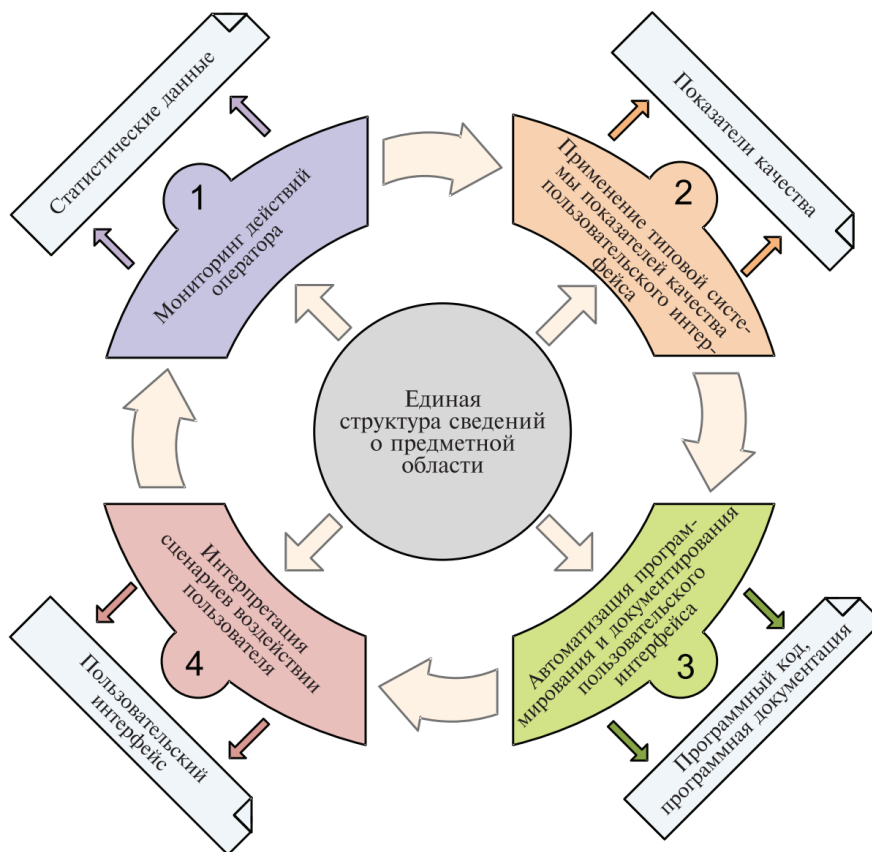


Рис. 4. Составные элементы подхода к созданию универсального средства построения пользовательского интерфейса программных средств

Мониторинг действий оператора (блок 1 на рис. 4) позволяет осуществлять сбор и накопление статистики деятельности оператора во время эксплуатации программных средств. Оператор вводит входные параметры с помощью технических средств ввода данных: клавиатуры, манипулятора «мышь», фотокамеры, микрофона, сканера, специализированных панелей кнопок и переключателей, внешних носителей информации и т. п. Каждая атомарная операция основывается на низкоуровневых сигналах от средства ввода, преобразуемых программной средой в воздействия ввода оператора за счет событийной обработки.

Оценка качества пользовательского интерфейса обеспечивается за счет:

- применения типовой системы показателей качества (блок 2 на рис. 4);
- разработки методики оценки качества;
- выполнения мероприятий по оценке показателей качества.

Система показателей качества пользовательского интерфейса основывается на сведениях представленных типов:

- аппаратные события – события, которые возникают от действия всех технических средств при работе с периферийным оборудованием, а также вследствие использования каналов связи и удаленных подключений;
- события приложения – события, связанные с операциями получения, обработки и преобразования данных, обеспечения политики безопасности и уровней доступа, т. е. все эти события появляются в процессе функционирования программы в программной среде;
- пользовательские события – события, предусмотренные программой, происходят в результате воздействий пользователя на элементы управления интерфейса.

Величины, с помощью которых можно определить качество выполнения операции:

- среднее время выполнения операции;
- число ошибок (количество итераций), возникающих за период работы;
- среднее время возникновения ошибки.

Показатели качества пользовательского интерфейса выражаются в формате:

- время выполнения операций (действий);
- число действий, необходимых для выполнения операции;
- число атомарных операций, необходимых для выполнения действия;
- число пустых воздействий, выполненных оператором;
- число ошибочных атомарных операций.

“Автоматизация программирования и документирования пользовательского интерфейса” (блок 3 на рис. 4) подразумевает возможность автоматизированного документирования интерфейса программы.



Основные мероприятия жизненного цикла пользовательского интерфейса:

- задание требований;
- проектирование;
- разработка, программирование;
- оценка качества и испытания;
- документирование (описание).

Каждое мероприятие жизненного цикла интерфейса взаимосвязано с UML моделями, описывающими интерфейс, которые в ходе итерационного процесса разработки корректируются и используются для получения документов.

В соответствии с подходом предусматривается автоматизация программирования макетов пользовательского интерфейса, основанная на таком выборе паттерна проектирования программного кода, который позволит инкапсулировать механизмы обработки данных и управления от элементов управления интерфейса, но при этом обеспечит связь с моделью данных и мониторинг действий оператора.

Можно использовать UML модель сценариев применения пользовательского интерфейса (модель сценариев воздействий пользователя (СВП)), описывающая различные стороны функционирования программы, которая выражает на определенном уровне абстракции порядок взаимодействия пользователя с программным средством. Модель используется для декомпозиции и формирования однозначного понимания сведений по совокупности функций и режимов работы разрабатываемого программного средства, а также для обеспечения возможности автоматизированного документирования и построения кода интерфейса программы.

С помощью UML модели СВП решаются следующие задачи:

- прототипирование и/или макетирование пользовательского интерфейса при разработке программного средства;
- формализация существующего пользовательского интерфейса;
- описание деятельности пользователя при эксплуатации программных средств, выраженное в виде набора осуществленных сценариев воздействий на элементы управления программы.

Отображение некоторого абстрактного сценария осуществляет механизм его интерпретации (блок 4 на рис. 4) в стандартные программные процедуры, характерные для выбранной программноаппаратной платформы. Наличие интерпретатора предписывает применение некоторой формальной логики (языка), с помощью лексем которой выражаются любые сценарии. Одна лексема описывает типовую атомарную операцию над элементом управления пользовательского интерфейса, идентифицирующие сведения о которой содержатся в параметрах лексемы.

Каждая атомарная операция, которая вызвана воздействиями пользователя, на диаграмме СВП представляет собой дугу графа, вершины

графа — элементы управления пользовательского интерфейса, веса графа — комплексные показатели, характеризующие:

- количество выполнений атомарной операции;
- время выполнения атомарной операции;
- количество ошибок, связанных с выполнением атомарной операции.

### **1.3 Построение пользовательского интерфейса с использованием интерактивного машинного обучения**

В процессе изучения статьи [5] были выделены этапы построения GUI с использованием интерактивного машинного обучения.

На первом этапе производится сбор входных данных. В качестве таких данных будут выступать частота, последовательность, достигаемый результат и время между применениями рассматриваемых функций. Исследованием в данной области занимается человеко-компьютерное взаимодействие, где в настоящее время основную роль играет машинное обучение.

На основании собранных данных проводится обучение, целью которого является сократить путь для достижения конкретного результата, сокращение количества шагов и затрачиваемого времени для выполнения идентичных задач. Для обучения используется алгоритм дерева градиентного повышения.

Алгоритм обучения выбирает лучший порог для каждого. Использование матрицы Гесса и весов позволяет вычислять прирост информации, вызванный применением каждой функции и правила принятия решения для узла.

Обучение может производиться на любом приложении с графическим пользовательским интерфейсом, имеющем длинные цепочки выполнения действий, например, пакет офисных приложений. По результатам обучения строится последовательность действий для достижения необходимого результата. При её построении учитывается время поиска элемента интерфейса, его доступность (подмножество действий необходимых к выполнению для получения доступа к данному элементу), соответствие описания и ожидаемого результат (использование других элементов, требующих большего числа шагов для достижения результата).

На основании полученных результатов вносятся корректировки в существующий интерфейс, после чего обучение продолжается.

## 2 Разработка web-приложения

### 2.1 Используемые программные решения

В рамках курсового проекта использовались Django, Docker и Nginx.

Django [6] — это высокоуровневый Python веб-фреймворк для бэкенда, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты. Фреймворк — это программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Главная цель фреймворка Django — позволить разработчикам вместо того, чтобы снова и снова писать одни и те же части кода, сосредоточиться на тех частях своего приложения, которые являются новыми и уникальными для их проекта.

Достоинства Django:

- Масштабируемый. Django использует компонентную архитектуру, то есть каждая её часть независима от других и, следовательно, может быть заменена или изменена, если это необходимо. Чёткое разделение частей означает, что Django может масштабироваться при увеличении трафика, путём добавления оборудования на любом уровне;
- Разносторонний. Django может быть использован для создания практически любого типа веб-сайтов;
- Безопасный. Django помогает разработчикам избежать многих распространённых ошибок безопасности, предоставляя фреймворк, разработанный чтобы «делать правильные вещи» для автоматической защиты сайта. Например, Django предоставляет безопасный способ управления учётными записями пользователей и паролями, избегая распространённых ошибок, таких как размещение информации о сессии в файлы cookie, где она уязвима или непосредственное хранение паролей вместо хэша пароля;
- Переносным. Django написан на Python, который работает на многих платформах;
- Удобным в сопровождении. Код Django написан с использованием принципов и шаблонов проектирования, которые поощряют создание поддерживаемого и повторно используемого кода.

Docker [7] — программное обеспечение с открытым исходным кодом, применяемое для разработки, тестирования, доставки и запуска веб-приложений в средах с поддержкой контейнеризации. Он нужен для более эффективного использования системы и ресурсов, быстрого развертывания готовых программных продуктов, а также для их масштабирования и переноса в другие среды с гарантированным сохранением стабильной работы.

Основной принцип работы Docker — контейнеризация приложений. Этот тип виртуализации позволяет упаковывать программное обеспечение по изолированным средам — контейнерам. Каждый из этих виртуальных блоков содержит все нужные элементы для работы приложения. Это дает возможность одновременного запуска большого количества контейнеров на одном хосте.

Достоинства использования Docker:

- Минимальное потребление ресурсов — контейнеры не виртуализируют всю операционную систему (ОС), а используют ядро хоста и изолируют программу на уровне процесса;
- Скоростное развертывание — вспомогательные компоненты можно не устанавливать, а использовать уже готовые docker-образы (шаблоны);
- Удобное скрывание процессов — для каждого контейнера можно использовать разные методы обработки данных, скрывая фоновые процессы;
- Работа с небезопасным кодом — технология изоляции контейнеров позволяет запускать любой код без вреда для ОС;
- Простое масштабирование — любой проект можно расширить, внедрив новые контейнеры;
- Удобный запуск — приложение, находящееся внутри контейнера, можно запустить на любом docker-хосте;
- Оптимизация файловой системы — образ состоит из слоев, которые позволяют очень эффективно использовать файловую систему.

Определения Docker:

- Docker-образ (Docker-image) — файл, включающий зависимости, сведения, конфигурацию для дальнейшего развертывания и инициализации контейнера;
- Docker-контейнер (Docker-container) — это легкий, автономный исполняемый пакет программного обеспечения, который включает в себя все необходимое для запуска приложения: код, среду выполнения, системные инструменты, системные библиотеки и настройки;
- Docker-файл (Docker-file) — описание правил по сборке образа, в котором первая строка указывает на базовый образ. Последующие команды выполняют копирование файлов и установку программ для создания определенной среды для разработки;
- Docker-клиент (Docker-client / CLI) — интерфейс взаимодействия пользователя с Docker-демоном. Клиент и Демон — важнейшие компоненты “движка” Докера (Docker Engine). Клиент Docker может взаимодействовать с несколькими демонами;
- Docker-демон (Docker-daemon) — сервер контейнеров, входящий в состав программных средств Docker. Демон управляет Docker-объектами (сети,

хранилища, образы и контейнеры). Демон также может связываться с другими демонами для управления сервисами Docker;

- Том (Volume) — эмуляция файловой системы для осуществления операций чтения и записи. Она создается автоматически с контейнером, поскольку некоторые приложения осуществляют сохранение данных;
- Реестр (Docker-registry) — зарезервированный сервер, используемый для хранения docker-образов;
- Docker-хаб (Docker-hub) или хранилище данных — репозиторий, предназначенный для хранения образов с различным программным обеспечением. Наличие готовых элементов влияет на скорость разработки;
- Docker-хост (Docker-host) — машинная среда для запуска контейнеров с программным обеспечением;
- Docker-сети (Docker-networks) — применяются для организации сетевого интерфейса между приложениями, развернутыми в контейнерах.

## 2.2 Разработка тестового web-приложения

На рис. 5 представлено содержание проекта.

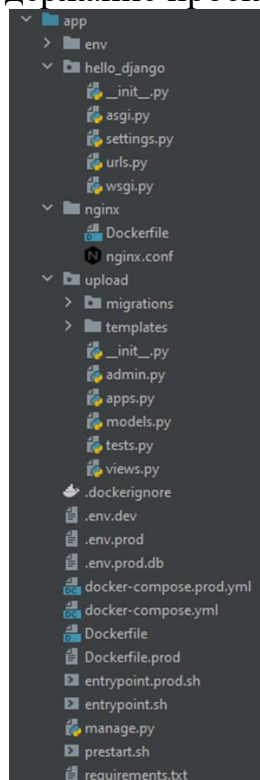


Рис. 5. Содержания проекта

- Файл settings.py содержит в себе все настройки проекта. Здесь регистрируются приложения, задаётся размещение статичных файлов, настройки базы данных и т.д.

- Файл `urls.py` задаёт ассоциации URL (унифицированный указатель ресурса) адресов с представлениями. Функция представления — это функция Python, которая принимает веб-запрос и возвращает веб-ответ. В случае данной работы ответом будет HTML-содержимое веб-страницы.
- `wsgi.py` используется для определения связи между Django приложением и веб-сервером.
- `manage.py` используется для создания приложений, работы с базами данных и для запуска отладочного сервера.
- В файле `Dockerfile.prod` описана последовательность команд, которые надо выполнить. На основе этого файла будет создан образ (docker image).

В `Dockerfile.prod` используется многоступенчатая сборка (multi-stage build), чтобы уменьшить конечный размер образа. `builder` — это временный образ, которое используется для сборки Python. Затем он копируется в конечный производственный образ, а образ `builder` отбрасывается. Так же был создан пользователь `app` без полномочий `root`. По умолчанию Docker запускает контейнерные процессы как `root` внутри контейнера, и если кто-то получит доступ на сервер и к контейнеру, то проникший на сервер пользователь тоже станет `root`, что, очевидно, не желательно. Таким образом увеличивается безопасность.

Вначале указан образ, на котором будем основываться. В нашем случае `python 3.9`. Устанавливается рабочая директория в контейнере с помощью `WORKDIR`. Далее устанавливаются переменные окружения:

- `PYTHONDONTWRITEBYTECODE` означает, что Python не будет пытаться создавать файлы `.pyc`;
- `PYTHONUNBUFFERED` гарантирует, что вывод консоли выглядит знакомым и не буферизируется Docker.

Затем команды для установки соответствующих пакетов, необходимых для `Psycopg2`. Из директории, где находится `Dockerfile.prod`, копируется файл зависимостей `requirements.txt` в рабочую директорию контейнера. Далее с помощью команды `RUN` исполняются перечисленные в ней команды, что приводит к установкам зависимостей. Затем копируется вся директория проекта в контейнер.

Были созданы папки `staticfiles` и `mediafiles`, так как `docker-compose` монтирует именованные тома как `root`. И так как используется пользователь `app` не обладающий полномочиями `root`, таким образом можно получить ошибку отказа в разрешении при запуске команды `collectstatic`, если каталог еще не существует.

#### Листинг 1. `Dockerfile.prod`

```
# BUILDER
FROM python:3.9.6-alpine as builder

# установка рабочей директории
```

```

WORKDIR /usr/src/app

# установка переменных окружения
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

# установите зависимости psycpg2
RUN apk update \
    && apk add postgresql-dev gcc python3-dev musl-dev

RUN pip install --upgrade pip
RUN pip install flake8
COPY . .
#RUN flake8 --ignore=E501,F401 .

# установка зависимостей
COPY ./requirements.txt .
RUN pip wheel --no-cache-dir --no-deps --wheel-dir
/usr/src/app/wheels -r requirements.txt

# FINAL

FROM python:3.9.6-alpine

# создание каталога для пользователя app
RUN mkdir -p /home/app

# создание пользователя app
RUN addgroup -S app && adduser -S app -G app

# создание необходимых директорий
ENV HOME=/home/app
ENV APP_HOME=/home/app/web
RUN mkdir $APP_HOME
RUN mkdir $APP_HOME/staticfiles
RUN mkdir $APP_HOME/mediafiles
WORKDIR $APP_HOME

# установка зависимостей
RUN apk update && apk add libpq
COPY --from=builder /usr/src/app/wheels /wheels
COPY --from=builder /usr/src/app/requirements.txt .
RUN pip install --no-cache /wheels/*

# копирование entrypoint.prod.sh

```

```

COPY ./entrypoint.prod.sh .
RUN sed -i 's/\r$//g' $APP_HOME/entrypoint.prod.sh
RUN chmod +x $APP_HOME/entrypoint.prod.sh

# копирование проекта в контейнер
COPY . $APP_HOME

# chown файлам пользователя
RUN chown -R app:app $APP_HOME

# переход к пользователю app
USER app

# запуск entrypoint.prod.sh
ENTRYPOINT ["/home/app/web/entrypoint.prod.sh"]

```

Был создан файл `docker-compose.prod.yml` (листинг 2). `docker-compose` позволяет управлять многоконтейнерностью. То есть можно запустить сразу несколько контейнеров, которые будут работать между собой. В нашем случае мы создаем контейнер с базой данных `db`, наш основной контейнер `web` и `nginx`.

Листинг 2. `docker-compose.prod.yml`

```

version: '3.8'

services:
  web:
    build:
      context: ./
      dockerfile: Dockerfile.prod
    command: gunicorn hello_django.wsgi:application --
bind 0.0.0.0:8080
    volumes:
      - static_volume:/home/app/web/staticfiles
      - media_volume:/home/app/web/mediafiles
    env_file:
      - ./env.prod
    depends_on:
      - db
  db:
    image: postgres:13.0-alpine
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    env_file:
      - ./env.prod.db
  nginx:

```



```

build: ./nginx
volumes:
  - static_volume:/home/app/web/staticfiles
  - media_volume:/home/app/web/mediafiles
ports:
  - 8084:80
  - 443:443
depends_on:
  - web

```

```

volumes:
  postgres_data:
  static_volume:
  media_volume:

```

В самом начале указывается версия docker-compose. Далее указываются services (контейнеры).

- web. Сбор проекта (build) делается на основе Dockerfile. Далее указывается команда для запуска сервера. Указываются volumes, что значит, что всё из указанных директорий используется в контейнере. Далее указываются порты (сервер Django запускается в контейнере на порте 8084 и этот порт перебрасывается на нашу хост машину). И в конце сервиса web указывается зависимость от сервиса db, то есть web не сможет работать без db.
- db. Выбирается образ postgres. Далее указываются volumes, чтобы сохранять наши данные. Также настраиваются переменные среды.
- nginx. Чтобы он действовал как обратный прокси-сервер для обработки клиентских запросов, а также для обслуживания статических файлов. Для работы Nginx была создана новая директория с соответствующим названием. В данной директории были созданы файлы Dockerfile (листинг 3) и nginx.conf (листинг 4).

#### Листинг 3. Dockerfile в директории nginx

```

FROM nginx:1.21-alpine
RUN rm /etc/nginx/conf.d/default.conf
COPY nginx.conf /etc/nginx/conf.d
# удаляем настройки по умолчанию
# копируем наши настройки

```

#### Листинг 4. nginx.conf

```

server {
    listen 80;
    location /static/ {
        alias /home/app/web/staticfiles/;
    }
}

```

```

    }
    location /media/ {
        alias /home/app/web/mediafiles/;
    }
    location / {
        proxy_pass http://web:8080;
        proxy_set_header Host $http_host;
        proxy_set_header Connection "upgrade";
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header Upgrade $http_upgrade;
        proxy_connect_timeout      600;
        proxy_send_timeout          600;
        proxy_read_timeout          600;
        send_timeout                 600;
    }
}

```

Команды, представленные на листинге 5, позволяют получить переменные окружения записанные в файл `.env.prod` (листинг 7) и записать их в переменные в файле проекта `settings.py`.

#### Листинг 5. Обновлённые переменные в `settings.py`

```

SECRET_KEY = os.environ.get("SECRET_KEY")

DEBUG = int(os.environ.get("DEBUG", default=0))

ALLOWED_HOSTS =
os.environ.get("DJANGO_ALLOWED_HOSTS").split(" ")

```

`SECRET_KEY` — это секретный ключ, который используется Django для поддержки безопасности сайта.

`DEBUG`. Включает подробные сообщения об ошибках, вместо стандартных HTTP статусов ответов. Должно быть изменено на `False` на сервере, так как эта информация очень много расскажет взломщикам.

`ALLOWED_HOSTS` — это список хостов/доменов, для которых может работать текущий сайт.

Так же в `setings.py` были обновлены настройки базы данных (листинг 6) на основе переменных окружения, определённых в `.env.prod`.

#### Листинг 6. Обновлённые переменные в `settings.py`

```

DATABASES = {
    "default": {

```

```

        "ENGINE": os.environ.get("SQL_ENGINE",
"django.db.backends.sqlite3"),
        "NAME": os.environ.get("SQL_DATABASE",
BASE_DIR / "db.sqlite3"),
        "USER": os.environ.get("SQL_USER", "user"),
        "PASSWORD": os.environ.get("SQL_PASSWORD",
"password"),
        "HOST": os.environ.get("SQL_HOST",
"localhost"),
        "PORT": os.environ.get("SQL_PORT", "5432"),
    }
}

```

### Листинг 7. Содержимое файла .env.prod

```

DEBUG=0
SECRET_KEY=change_me
DJANGO_ALLOWED_HOSTS=localhost 195.19.40.68 [::1]
SQL_ENGINE=django.db.backends.postgresql
SQL_DATABASE=hello_django_prod
SQL_USER=hello_django
SQL_PASSWORD=hello_django
SQL_HOST=db
SQL_PORT=5432
DATABASE=postgres

```

Для работы с медиафайлами был создан новый модуль Django под названием upload, то есть была создана новая директория upload (новый модуль создаётся командой в терминале: “ docker-compose exec web python manage.py startapp upload”) и добавлен новый модуль в INSTALLED\_APPS в settings.py.

В созданной директории был изменен файл views.py (листинг 9), была создана папка для шаблонов “templates” и в ней был создан новый шаблон title.html (листинг 10). Так же был изменен файл app/hello\_django/urls.py (листинг 8).

### Листинг 8. urls.py

```

from django.contrib import admin
from django.urls import path
from django.conf import settings
from django.conf.urls.static import static
from upload.views import image_upload
urlpatterns = [
    path("", gui, name="gui"),
    path("image_upload/", image_upload, name="upload"),
    path("admin/", admin.site.urls),]

```

```

if bool(settings.DEBUG):
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)

```

#### Листинг 9. views.py

```

from django.shortcuts import render
from django.core.files.storage import FileSystemStorage

def gui(request):
    return render(request, "title.html")

```

#### Листинг 10. title.html

```

<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>title</title>
</head>
<body>
<h1>Пункт 1</h1>
<p><b>Параметр X</b><br>
<input type="text" value="VX"></p>
<p><b>Параметр Y</b><br>
<input type="text" value="YX"></p>
<p><<input type="checkbox" name="Флажок 1" checked>Флажок
1</p>
<p><<input type="checkbox" name="Флажок 2" checked>Флажок
2</p>
<h1>Пункт 2</h1>
<p><<input type="checkbox" name="Флажок 3">Флажок 3</p>
</body>
</html>

```

## 2.3 Запуск web-приложения на тестовом сервере

После входа на сервер через консоль и запуска приложения, при переходе по <http://195.19.40.68:8084>, открывается страница (рис. 6).

# Пункт 1

Параметр X

Параметр Y

< ☒ Флажок 1

< ☒ Флажок 2

# Пункт 2

< ☐ Флажок 3

Рис. 6. Тестовая страница

Далее проверим работу базы данных. Входим в контейнер и создаём суперпользователя (рисунок 7).

```
PS D:\PyCharm\django-on-docker\app> docker-compose exec web sh
/usr/src/app # ls
Dockerfile          docker-compose-old.yml  entrypoint.prod.sh    hello_django          requirements.txt
Dockerfile.prod     docker-compose.prod.yml  entrypoint.sh         manage.py
catalog             docker-compose.yml      env                   nginx
/usr/src/app # python manage.py createsuperuser
Username (leave blank to use 'root'): Arthur
Email address:
Password:
Password (again):
Superuser created successfully.
/usr/src/app #
```

Рис.7 Создание суперпользователя

Переходим по адресу <http://195.19.40.68:8084/admin> и вводим имя и пароль суперпользователя. Вход выполнен (рисунок 8).

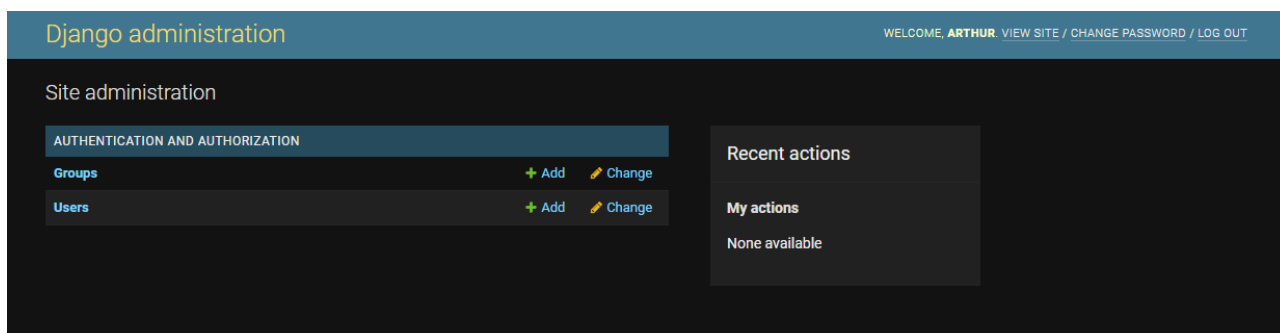


Рис. 8 Вход успешно выполнен

Проверка на создание таблиц по умолчанию на рисунке 9.

```
PS D:\PyCharm\django-on-docker\app> docker-compose exec db psql --username=hello_django --dbname=hello_django_dev
psql (13.0)
Type "help" for help.

hello_django_dev=# \l
```

List of databases					
Name	Owner	Encoding	Collate	Ctype	Access privileges
hello_django_dev	hello_django	UTF8	en_US.utf8	en_US.utf8	
postgres	hello_django	UTF8	en_US.utf8	en_US.utf8	
template0	hello_django	UTF8	en_US.utf8	en_US.utf8	=c/hello_django +
					hello_django=CtC/hello_django
template1	hello_django	UTF8	en_US.utf8	en_US.utf8	=c/hello_django +
					hello_django=CtC/hello_django

(4 rows)

Рис. 9. Проверка таблиц

## 3 Генерация GUI на основе aINI

### 3.1 Соответствие aINI кода с элементами интерфейса

Было реализована генерация следующих элементов интерфейса:

- **Переключаемая вкладка**

Описание в aINI: [sec1]//Пункт 1

Вывод в HTML на листинге 11.

Листинг. 11. HTML-код кнопки вкладки

```
<div class="tab">
  <button class="tablinks"
onclick="tabs(event, 'section_0') "
id="defaultOpen">Пункт 1</button>
</div>
```

Создаются кнопки для открытия содержимого определенной вкладки. Все элементы <div> с классом tabcontent скрыты по умолчанию с помощью CSS (каскадные таблицы стилей) (листинг 12) и JS (JavaScript) (листинг 16). Когда пользователь нажимает на кнопку - открывается содержимое вкладки, которое соответствует этой кнопке.

Листинг. 12. CSS стили

```
<style>
body {font-family: Arial;}

.tab {
  overflow: hidden;
  border: 1px solid #ccc;
  background-color: #f1f1f1;
}

.tab button, .clickable {
  float: left;
border: none;
  outline: none;
  cursor: pointer;
  padding: 14px 16px;
  transition: 0.3s;
  font-size: 17px;
}

.tab button:hover, .clickable:hover {
  background-color: #ddd;
```

```

}

.tab button.active {
    background-color: #ccc;
}

.tabcontent {
    display: none;
    padding: 6px 12px;
    border: 1px solid #ccc;
    border-top: none;
}
</style>

```

Получение всех элементов с классом tabcontent и скрытие их (листинг 13).

#### Листинг. 13. JavaScript код скрытия вкладок

```

tabcontent =
document.getElementsByClassName("tabcontent");
for (i = 0; i < tabcontent.length; i++) {
    tabcontent[i].style.display = "none";
}

```

Получение всех элементов с классом \textsf{tablinks} и удаления у них класса active (листинг 14).

#### Листинг. 14. JavaScript код удаления класса active

```

tablinks = document.getElementsByClassName("tablinks");
for (i = 0; i < tablinks.length; i++) {
    tablinks[i].className = tablinks[i].className.replace(" active", "");
}

```

Изображение текущей вкладки и добавление класса active кнопке, открывающую эту вкладку (листинг 15).

#### Листинг. 15. JavaScript код изображения текущей вкладки

```

document.getElementById(tab_id).style.display =
"block";
evt.currentTarget.className += " active";

```

Функция tabs (листинг 16) вызывается при событии onclick (нажатие) у элементов кнопок вкладок, например листинг 11.

#### Листинг. 16. полный JavaScript код

```

<script>

```



```
function tabs(evt, tab_id) {
    var i, tabcontent, tablinks;
    tabcontent =
document.getElementsByClassName("tabcontent");
    for (i = 0; i < tabcontent.length; i++) {
        tabcontent[i].style.display = "none";
    }
    tablinks =
document.getElementsByClassName("tablinks");
    for (i = 0; i < tablinks.length; i++) {
        tablinks[i].className =
tablinks[i].className.replace(" active", "");
    }
    document.getElementById(tab_id).style.display =
"block";
    evt.currentTarget.className += " active";
}

document.getElementById("defaultOpen").click();
</script>
```

- **Текст**

Входные aINI-данные: //здесь пишется текст

Выходные HTML-данные на листинге 17.

Листинг. 17. HTML-код текста

```
<p>здесь пишется текст</p>
```

- **Ссылка**

Входные aINI-данные: [https://bmstu.ru/about] //Дополнительная информация

Выходные HTML-данные на листинге 18.

Листинг. 18. HTML-код ссылки

```
<p><a href="https://bmstu.ru/about">Дополнительная
информация</a></p>
```

- **Поле ввода**

Входные aINI-данные: x=12//Параметр X

Если вместо значения по умолчанию записать “@переменная@” (в данном случае @x@), то поле будет пустым.

Выходные HTML-данные на листинге 19.

Листинг 19. HTML-код поля ввода

```
<p><b>Параметр X</b><br>
```

```
<input type="text" name="x" value="12"></p>
```

Часть функции представления, отвечающая за сохранение в исходный aINI-файл записанных в данное поле ввода значений, на листинге 20.

Листинг. 20. Часть функции представления для поля ввода

```
if request.method == "POST":
    if request.POST["x"]:
        x = request.POST["x"]
        new_data = ''
        input = open('/home/app/web/input/input1',
'r', encoding='utf-8')
        input_f = File(input)
        old_data = input_f.readlines()
        for l in old_data:
            if l.find('//Параметр X\n') != -1:
                l = 'x=' + x + '//Параметр X\n'
                new_data = new_data + str(l)
        output = open('/home/app/web/input/input1',
'w', encoding='utf-8')
        output_f = File(output)
        output_f.write(new_data)
        input_f.close()
        input.close()
        output_f.close()
        output.close()
```

- **Флажок (переключатель)**

Входные aINI-данные: box1=[1]{0|1} // Флажок 1

В квадратных скобках указывается значение по умолчанию. «1» – флажок будет активен по умолчанию с помощью добавления к HTML-коду атрибута checked, «0» - флажок будет неактивен.

Выходные HTML-данные на листинге 21.

Листинг. 21. HTML-код флажка

```
<p><input type="checkbox" name="box1" value="box1"
checked>Флажок 1</p>
```

Часть функции представления, отвечающая за сохранение в исходный aINI-файл изменений состояния флажка, на листинге 22.

Листинг. 22. Часть функции представления для флажка

```
if "box1" in request.POST:
    box1 = '1'
else:
    box1 = '0'
```

```

new_data = ''
input = open('/home/app/web/input/input1', 'r',
encoding='utf-8')
input_f = File(input)
old_data = input_f.readlines()
for l in old_data:
    if l.find('//Флажок 1\n') != -1:
        l = 'box1=[' + box1 + ']{0|1}//Флажок
1\n'
        new_data = new_data + str(l)
output = open('/home/app/web/input/input1', 'w',
encoding='utf-8')
output_f = File(output)
output_f.write(new_data)
input_f.close()
input.close()
output_f.close()
output.close()

```

- **Поле выбора файла**

Входные aINI-данные: ParametersFile=[file]//Выберите  
Выходные HTML-данные на листинге 23.

Листинг. 23. HTML-код поля выбора файла

```

<p><b>Выберите файл</b><br>
<input type="file" name="ParametersFile"></p>

```

Часть функции представления, отвечающая за сохранение в папку с входными данными файла, выбранного пользователем, на листинге 24.

Листинг. 24.

```

if "ParametersFile" in request.FILES:
    ParametersFile =
request.FILES["ParametersFile"]
    fs = FileSystemStorage()
    filename = fs.save(ParametersFile.name,
ParametersFile)
    file_url = fs.url(filename)

```

## 3.2 Разработка программы

Во время разработки программы преобразования данных формата aINI в HTML-код использовался модуль синтаксического анализа для языка Python – Pyparsing[8].

После запуска программа в функции main (листинг 25) вначале открывается файл config с входными данными и создаются файлы для записи выходных данных. Далее считываются построчно данные из файла config (пример содержимого файла config на листинге 26). В данном файле представлены в квадратных скобках названия искомых файлов, на основе которых будут генерироваться HTML-страницы (содержимое такого файла представлено на листинге 27), до знака "=" - URL для соответствующей страницы, а после "/" наименование страницы. На основе файла config также будет создана страница menu (пример меню – листинг 32), из которой можно будет перейти на все страницы, перечисленные в config. В функции main создаются ещё два текстовых файла: txt\_for\_urls, txt\_for\_views. В этих файлах будет записан дополнительный код, который пользователю необходимо будет вставить в соответствующие файлы в web-приложения для корректного взаимодействия приложения с HTML-файлами (в комментариях в файлах присутствует информация о том, куда должен быть скопирован код). В файле txt\_for\_urls.txt записывается код для файла urls.py web-приложения, то есть элементы массива urlpatterns (список всех URL, которые обрабатываются web-приложением). В файле txt\_for\_views записывается код для файла views.py web-приложения, то есть код функций представления (функция, которая обрабатывает запрос по соответствующему URL адресу) для каждого обрабатываемого URL. Пример файлов txt\_for\_urls и txt\_for\_views представлен на листингах 32 и 33. Наименование страницы и название файла с данными формата aINI считанные из config передаются функции aini\_to\_html (листинг 28).

#### Листинг 25. Функция main

```
def main():
    input_f = open(r'input\config', encoding='utf-8')
    f_menu = open(r'output\menu.html', 'w',
encoding='utf-8')
    f_urls = open(r'output\txt_for_urls.txt', 'w',
encoding='utf-8')
    f_views = open(r'output\txt_for_views.txt', 'w',
encoding='utf-8')
    list_of_functions = 'from upload.views import
menu'

    f_urls.write('#Ниже представлен код, который
должен быть вставлен в файл urls.py в список
urlpatterns\n\n\tpath('
        '""', menu, name="menu"),\n')
    f_views.write(
        '#Ниже представлен код, который должен быть
вставлен в файл views.py\n\n\ndef
menu(request):\n\n\treturn render('
```

```

        'request, "menu.html")\n\n')
    f_menu.write(
        '<!DOCTYPE                                html>\n<html
lang="en">\n<head>\n<meta charset="UTF-8">\n<style>\nbody
{font-family: '
        'Arial;}\n\n.tab                                {\n\toverflow:
hidden;\n\tborder: 1px solid #ccc;\n\tbackground-color:
#f1f1f1;\n}\n\n.tab '
        'button {\n\toverflow: hidden;\n\tborder: 1px
solid #ccc;\n\tbackground-color: #ddd;\n\tfloat: '
        'left;\n\tborder:                                none;\n\toutline:
none;\n\tcursor:                                pointer;\n\tpadding:                                14px
16px;\n\ttransition: '
        '0.3s;\n\tfont-size: '
        '17px;\n\tdisplay:                                block;\n\twidth:
100%;\n}\n\n.tab button:hover {\n\tbackground-color: '
        '#808080;\n}\n\n</style>\n<title>menu</title>\n</head>\n<
body>\n<h1>Здравствуй! Выберите один из '
        'предложенных                                вариантов.</h1>\n<div
class="tab">\n')

```

```

    for line in input_f:
        global num_of_section
        num_of_section = -1

        try:
            config                                =
            aini_file_name.parseString(line).asList()
        except ParseException:
            print('The string is not parsed:')
            print(line)
            quit()

        f_menu.write('\t<button
onclick="document.location=\'/' + config[0] + '\\'>' +
config[6] + '</button>\n')
        f_urls.write('\t\tpath("'" + config[0] + "'", '
+ config[3] + "'", name='" + config[3] + "'),\n')
        list_of_functions = list_of_functions + ', '
+ config[3]
        list_of_variables = aini_to_html(config[3],
config[6], )
        f_views.write('\n\ndef ' + config[3] +
'(request):\n\tif request.method == "POST":\n')

```

```

        for variable in list_of_variables:
            f_views.write(variable)
        f_views.write('\t\treturn render(request, "'
+ config[3] + '.html")\n\treturn render(request, "' +
config[
        3] + '.html")\n\n')

    f_menu.write('</div>\n</body>\n</html>\n')
    f_urls.write('\n#Также вставьте следующую строку
в файл urls.py до списка urlpatterns\n' +
list_of_functions)

    input_f.close()
    f_menu.close()
    f_urls.close()
    f_views.close()

```

Листинг. 26. Пример файла config

```

F1 = [input1]//Test1
F2 = [input2]//Test2
F3 = [input3]//Test3

```

Листинг. 27. Пример полученного input1

```

[sec1]//Вкладка 1
x=25//Параметр X
y=@y@//Параметр Y
box1=[0]{0|1}//Флажок 1
box2=[1]{0|1}//Флажок 2
[sec2]//Вкладка 2
q=ABC//Параметр Q
box3=[0]{0|1}//Флажок 3
ParametersFile=[file]//Выберите требуемый файл
//Московский государственный технический университет им.
Н. Э. Баумана - фкпфроссийский национальный
исследовательский университет, научный центр, особо
ценный объект культурного наследия народов России.
[https://bmstu.ru/about]//Дополнительная информация

```

В функции `aini_to_html`(листинг 28) построчно считываются данные из aINI файла и происходит распознавание строки в соответствии с шаблонами в парсере (листинг 29). Содержимое каждой вкладки формы (то есть элементы, описанные между вкладками в aINI файле) записывается последовательно в один элемент списка `elements_list`, а сами вкладки записываются в список `sections_list`. При использовании `ruparsing`, парсер вначале был написан для отдельных ключевых элементов (например, числовые значения, текст из

английских и русских букв, ссылка и т. п.), а потом из отдельных частей получается парсер для всей строки aINI-кода.

Распознавание строк происходит в функции parsing (листинг 30), которая возвращает список из распознанных частей строки и номер распознанного элемента интерфейса для дальнейшей генерации. Распознавание строки происходит в блоке с помощью функции parseString(s).asList() модуля ryparsing. Данная функция получает на вход строку s и проводит синтаксический разбор данной строки по шаблону, через который вызвана функция (например, в случае parse\_file\_selection.parseString(s).asList() строка будет разобрана как строка элемента выбора файла). Если разбор был выполнен без возврата исключения (тип данных в python, сообщающий программисту об ошибках [9]), то в возвращаемые переменные записываются соответствующие значения, иначе благодаря конструкции try-except происходит перехват исключения.

На основе списка и номера элемента интерфейса создается HTML-код с помощью вызова внутри функции aini\_to\_html соответствующей функции для каждого из предусмотренных элементов интерфейса (листинг 31). Для некоторых элементов интерфейса (окно для ввода текста, флажок, выбор файла и т. п.) функция создания HTML-кода возвращает код для функции представления, который далее будет выведен в ранее упомянутый файл txt\_for\_views.

Сгенерированные функции представления (листинг 33) получают данные из форм ввода, записанные после взаимодействия с пользователем, после чего с помощью функции find производится поиск строки элемента формы ввода в файле, описывающем всю страницу, с которой взаимодействовал пользователь. Новые значения записываются вместо старых значений. В случае элемента выбора файла, выбранный пользователем файл сохраняется в папку, где находятся файлы с входными данными.

Листинг. 28. Функция aini\_to\_html

```
def aini_to_html(file_name, title):
    sections_list = []
    elements_list = []
    list_of_variables = []

    input_f = open('input/' + file_name, encoding='utf-
8')
    f = html_start(title, file_name)

    for line in input_f:
        id_and_line = parsing(line)
        patern_id = id_and_line[0]
        parsed_line = id_and_line[1]
        print(parsed_line)
```

```

        if patern_id <= 3:
            match patern_id:
                case 1:
                    html_code =
section_to_html(parsed_line)
                    sections_list.append(html_code)
                case 2:
                    html_code =
textbox_to_html(parsed_line, file_name)
                    if len(elements_list) ==
num_of_section + 1:
                        elements_list[num_of_section] +=
html_code[1]
                    else:

elements_list.append(html_code[1])

list_of_variables.append(html_code[0])
                case 3:
                    html_code = box_to_html(parsed_line,
file_name)
                    if len(elements_list) ==
num_of_section + 1:
                        elements_list[num_of_section] +=
html_code[1]
                    else:

elements_list.append(html_code[1])

list_of_variables.append(html_code[0])
            else:
                match patern_id:
                    case 4:
                        html_code =
file_selection_to_html(parsed_line)
                        if len(elements_list) ==
num_of_section + 1:
                            elements_list[num_of_section] +=
html_code[1]
                        else:

elements_list.append(html_code[1])

list_of_variables.append(html_code[0])
                    case 5:

```



```

        html_code = link_to_html(parsed_line)
        if len(elements_list) ==
num_of_section + 1:
            elements_list[num_of_section] +=
html_code
        else:
            elements_list.append(html_code)
    case 6:
        html_code = text_to_html(parsed_line)
        if len(elements_list) ==
num_of_section + 1:
            elements_list[num_of_section] +=
html_code
        else:
            elements_list.append(html_code)

    for section in sections_list:
        f.write(section)
    f.write('</div>\n\n')

    i = 0
    f.write('<form method="post" enctype="multipart/form-
data">\n\n{% csrf_token %}\n')
    for element in elements_list:
        id = 'section_' + str(i)
        f.write('<div id="' + id + '"
class="tabcontent">\n' + element + '</div>\n\n')
        i += 1
    html_finish(f)
    return list_of_variables

```

### Листинг. 29. Парсер

```

rus_alphas='ёйцукенгшщзхъфывапролджэячсмитьбюЁЙЦУКЕНГШЩЗХ
ЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ'

```

```

bool_var = Word('0' + '1')
word_num = Word(nums + alphas + '@')
variable = Word(nums + alphas + '_')
rus_eng_word_num = Word(alphas + rus_alphas + ' ' + nums)
link = Word(alphanums + '-' + '+' + '=' + '/' + '.' + '_'
+ '#' + ':' + '&' + '?' + '%')
text = Word(printables + rus_alphas + ' ')
menu_link = Word(alphanums + '-' + '+' + '=' + '.' + '_'
+ '#' + ':' + '&' + '?' + '%')

```

```

parse_section = '[' + variable + ']' + '//' +
rus_eng_word_num
parse_text_box = variable + '=' + word_num + '//' +
rus_eng_word_num
parse_box = variable + '=' + '[' + bool_var + ']' +
'{0|1}' + '//' + rus_eng_word_num
parse_file_selection = variable + '=' + '[file]' + '//' +
rus_eng_word_num
parse_link = '[' + link + ']' + '//' + rus_eng_word_num
parse_text = '//' + OneOrMore(text)
aini_file_name = menu_link + '=' + '[' + variable + ']' +
'//' + rus_eng_word_num

```

### Листинг. 30. Функция parsing

```

def parsing(s):
    try:
        test = parse_section.parseString(s).asList()
        patern_id = 1
    except ParseException:
        try:
            test = parse_text_box.parseString(s).asList()
            patern_id = 2
        except ParseException:
            try:
                test = parse_box.parseString(s).asList()
                patern_id = 3
            except ParseException:
                try:
                    test =
parse_file_selection.parseString(s).asList()
                    patern_id = 4
                except ParseException:
                    try:
                        test =
parse_link.parseString(s).asList()
                        patern_id = 5
                    except ParseException:
                        try:
                            test =
parse_text.parseString(s).asList()
                            patern_id = 6
                        except ParseException:

```

```

print('The string is not
parsed:' + s + ')
quit()
return [patern_id, test]

```

### Листинг. 31. Функции для записи HTML-кода

```

def html_start(title, html_file_name):
    folder = 'output\\'
    f = open(folder + html_file_name + '.html', 'w',
encoding='utf-8')
    f.write('<!DOCTYPE html>\n<html>\n<head>\n<meta
name="viewport" content="width=device-width, '
'initial-scale=1">\n<style>\nbody {font-
family: Arial;}\\n\\n.tab {\\n\\toverflow: hidden;\\n\\tborder:
1px '
'solid #ccc;\\n\\tbackground-color:
#f1f1f1;\\n}\\n\\n.tab button, .clickable {\\n\\tfloat:
left;\\nborder: '
'none;\\n\\toutline: none;\\n\\tcursor:
pointer;\\n\\tpadding: 14px 16px;\\n\\ttransition:
0.3s;\\n\\tfont-size: '
'17px;\\n}\\n\\n.tab button:hover,
.clickable:hover {\\n\\tbackground-color: #ddd;\\n}\\n\\n.tab
,
'button.active {\\n\\tbackground-color:
#ccc;\\n}\\n\\n.tabcontent {\\n\\tdisplay: none;\\n\\tpadding:
6px '
'12px;\\n\\tborder: 1px solid #ccc;\\n\\tborder-
top: none;\\n}\\n</style>\n<title>' + title +
'</title>\n</head>\n<body>\n \\n<div
class="tab">\n')
    return f

def section_to_html(parsed_line):
    if len(parsed_line) != 5:
        section = parsed_line[1]
    else:
        section = parsed_line[4]
    global num_of_section
    num_of_section += 1
    id = "section_" + str(num_of_section) + ""
    if num_of_section == 0:

```



```
+ parsed_line[0] +
'"']\n\t\t\tnew_data = \'\' \n\t\t\tinput =
open(\'/home/app/web/input/'
+ file_name + '\', \'r\',
encoding=\'utf-8\')\n\t\t\tinput_f =
File(input)\n\t\t\told_data = '

'input_f.readlines()\n\t\t\tfor l in
old_data:\n\t\t\t\tif l.find(\'//\' +
+ parsed_line[4] + '\\n\') != -
1:\n\t\t\t\t\tl = \'\' + parsed_line[0] + '=\' + ' +
parsed_line[0]
+ ' + \'//\' + parsed_line[4] +
'\n\''\n\t\t\t\tnew_data = new_data +
str(l)\n\t\t\t\toutput = '

'open(\'/home/app/web/input/' + file_name +
+ '\', \'w\', encoding=\'utf-
8\')\n\t\t\t\toutput_f =
File(output)\n\t\t\t\toutput_f.write('

'new_data)\n\t\t\t\tinput_f.close()\n\t\t\t\tinput.close()\n\t\t\t\toutput_f.close('
+)\n\t\t\t\toutput.close()\n\n')
print(parsed_line)

if parsed_line[2] == '@' + parsed_line[0] + '@':
    value = ''
else:
    value = parsed_line[2]
    html_code.append('\t<p><b>' + textbox +
'</b><br>\n\t<input type="text" name="' + parsed_line[0]
+ '" value='
+ value + '></p>\n')

return html_code

def box_to_html(parsed_line, file_name):
    html_code = []
    if len(parsed_line) != 8:
        box = parsed_line[0]
        html_code.append('\t\tif "' + parsed_line[0] + '"
in request.POST:\n\t\t\t' + parsed_line[0] +
```

```

        ' = \'1\'\\n\\t\\telse:\\n\\t\\t\\t' +
parsed_line[0] +
        ' = \'0\'\\n\\t\\tnew_data =
\\'\'\\n\\t\\tinput = open(\\'/home/app/web/input/' +
file_name +
        '\\', \\r\\', encoding=\\'utf-
8\\')\\n\\t\\tinput_f = File(input)\\n\\t\\told_data = '
        'input_f.readlines()\\n\\t\\tfor l
in old_data:\\n\\t\\t\\tif l.find(\\'\\n' + parsed_line[0] +
        '\\') != -1:\\n\\t\\t\\t\\t1 = \\'' +
parsed_line[0] + '\\[\\' + ' + parsed_line[0] + ' +
\\']{0|1}//' +
        parsed_line[0] +
        '\\n\\'\\n\\t\\t\\tnew_data = new_data + str(l)\\n\\t\\toutput =
open('
        '\\'/home/app/web/input/' + file_name + '\\', \\w\\', '
        'encoding=\\'utf-8\\')\\n\\t\\toutput_f = '
        'File(output)\\n\\t\\toutput_f.write('
        'new_data)\\n\\t\\tinput_f.close('
        ')\\n\\t\\tinput.close('
        ')\\n\\t\\toutput_f.close('
        ')\\n\\t\\toutput.close()\\n\\n')
    else:
        box = parsed_line[7]
        html_code.append('\\t\\tif "' + parsed_line[0] + '"
in request.POST:\\n\\t\\t\\t' + parsed_line[0] +
        ' = \'1\'\\n\\t\\telse:\\n\\t\\t\\t' +
parsed_line[0] + '\\ = \'0\'\\n\\t\\tnew_data = \\'\'\\n\\t\\tinput
= '
        'open(\\'/home/app/web/input/' + file_name +
        '\\', \\r\\', encoding=\\'utf-
8\\')\\n\\t\\tinput_f = File(input)\\n\\t\\told_data = '
        'input_f.readlines()\\n\\t\\tfor l
in old_data:\\n\\t\\t\\tif l.find(\\'//' + parsed_line[7] +
        '\\n\\') != -1:\\n\\t\\t\\t\\t1 = \\''
+ parsed_line[0] + '\\[\\' + ' + parsed_line[0] + ' +
\\']{0|1}//'

```

```

        + parsed_line[7] +
'\n\\n\\n\t\t\tnew_data = new_data + str(l)\n\t\t\toutput =
open('

'\'/home/app/web/input/' + file_name + '\', \'w\','

"encoding=\'utf-8\')\n\t\t\toutput_f '

'= File('

'output)\n\t\t\toutput_f.write('

'new_data)\n\t\t\tinput_f.close('

')\n\t\t\tinput.close('

')\n\t\t\toutput_f.close('

')\n\t\t\toutput.close()\n\n')
    if parsed_line[3] == '1':
        html_code.append('\t<p><input type="checkbox"
name="" + parsed_line[0] + ' value="" + parsed_line[0]
        + ' checked>' + box + '</p>\n')
    else:
        html_code.append('\t<p><input type="checkbox"
name="" + parsed_line[0] + ' value="" + parsed_line[0] +
        '>'
        + box + '</p>\n')
    return html_code

def file_selection_to_html(parsed_line):
    html_code = []
    if len(parsed_line) != 5:
        file_selection = parsed_line[0]
    else:
        file_selection = parsed_line[4]
    html_code.append('\t\t\tif "' + parsed_line[0] + '" in
request.FILES:\n\t\t\t\t' + parsed_line[0] + ' =
request.FILES["'
        + parsed_line[0] + '"]\n\t\t\t\ttfs =
FileSystemStorage()\n\t\t\t\tfilename = fs.save(' +
parsed_line[0]
        + '.name, ' + parsed_line[0] +
')\n\t\t\t\t\tfile_url = fs.url(filename)\n\n')

```







## 4 Тестирование

Было произведено тестирование web-приложения с сгенерированным интерфейсом, элементами списка urlpatterns (листинг 34) и функциями представления (листинг 35). Примеры главного меню и одного из трех HTML-файлов представлены в листингах 32 и 33 соответственно, они были сгенерированы на основе aINI-кода из листингов 26 и 27.

Листинг. 32. menu.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<style>
body {font-family: Arial;}

.tab {
    overflow: hidden;
    border: 1px solid #ccc;
    background-color: #f1f1f1;
}

.tab button {
    overflow: hidden;
    border: 1px solid #ccc;
    background-color: #ddd;
    float: left;
    border: none;
    outline: none;
    cursor: pointer;
    padding: 14px 16px;
    transition: 0.3s;
    font-size: 17px;
    display: block;
    width: 100%;
}

.tab button:hover {
    background-color: #808080;
}

</style>
<title>menu</title>
</head>
<body>
```

```

<h1>Здравствуйте! Выберите один из предложенных
вариантов.</h1>
<div class="tab">
  <button
onclick="document.location='/F1/'">Test1</button>
  <button
onclick="document.location='/F2/'">Test2</button>
  <button
onclick="document.location='/F3/'">Test3</button>
</div>
</body>
</html>

```

### Листинг. 33. input1.html

```

<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,
initial-scale=1">
<style>
body {font-family: Arial;}

.tab {
  overflow: hidden;
  border: 1px solid #ccc;
  background-color: #f1f1f1;
}

.tab button, .clickable {
  float: left;
border: none;
  outline: none;
  cursor: pointer;
  padding: 14px 16px;
  transition: 0.3s;
  font-size: 17px;
}

.tab button:hover, .clickable:hover {
  background-color: #ddd;
}

.tab button.active {
  background-color: #ccc;
}

```

```

.tabcontent {
    display: none;
    padding: 6px 12px;
    border: 1px solid #ccc;
    border-top: none;
}
</style>
<title>Test1</title>
</head>
<body>

<div class="tab">
    <button class="tablinks"
onclick="tabs(event, 'section_0') "
id="defaultOpen">Вкладка 1</button>
    <button class="tablinks"
onclick="tabs(event, 'section_1') ">Вкладка 2</button>
</div>

<form method="post" enctype="multipart/form-data">

{% csrf_token %}
<div id="section_0" class="tabcontent">
    <p><b>Параметр X</b><br>
    <input type="text" name="x" value="25"></p>
    <p><b>Параметр Y</b><br>
    <input type="text" name="y" value=""></p>
    <p><input type="checkbox" name="box1"
value="box1">Флажок 1</p>
    <p><input type="checkbox" name="box2" value="box2"
checked>Флажок 2</p>
</div>

<div id="section_1" class="tabcontent">
    <p><b>Параметр Q</b><br>
    <input type="text" name="q" value="ABC"></p>
    <p><input type="checkbox" name="box3"
value="box3">Флажок 3</p>
    <p><b>Выберите требуемый файл</b><br>
    <input type="file" name="ParametersFile"></p>
    <p>Московский государственный технический университет
им. Н. Э. Баумана – фкпфроссийский национальный
исследовательский университет, научный центр, особо
ценный объект культурного наследия народов России.</p>

```

```

        <p><a href="https://bmstu.ru/about">Дополнительная
информация</a></p>
</div>

```

```

<p><input class="clickable" type="submit"></p>
</form>

```

```

<br>

```

```

<br>

```

```

<br>

```

```

<button class="clickable"
onclick="document.location='/'">Назад</button>

```

```

<script>

```

```

function tabs(evt, tab_id) {
    var i, tabcontent, tablinks;
    tabcontent =
document.getElementsByClassName("tabcontent");
    for (i = 0; i < tabcontent.length; i++) {
        tabcontent[i].style.display = "none";
    }
    tablinks =
document.getElementsByClassName("tablinks");
    for (i = 0; i < tablinks.length; i++) {
        tablinks[i].className =
tablinks[i].className.replace(" active", "");
    }
    document.getElementById(tab_id).style.display =
"block";
    evt.currentTarget.className += " active";
}

```

```

document.getElementById("defaultOpen").click();
</script>

```

```

</body>

```

```

</html>

```

### Листинг. 34. Пример файла txt\_for\_urls

#Ниже представлен код, который должен быть вставлен в файл urls.py в список urlpatterns

```

path("", menu, name="menu"),
path("F1/", input1, name="input1"),
path("F2/", input2, name="input2"),
path("F3/", input3, name="input3"),

```

```
#Также вставьте следующую строку в файл urls.py до списка
urlpatterns
from upload.views import menu, input1, input2, input3
```

### Листинг. 35. Пример файла txt\_for\_views

#Ниже представлен код, который должен быть вставлен в файл views.py

```
def menu(request):
    return render(request, "menu.html")

def input1(request):
    if request.method == "POST":
        if request.POST["x"]:
            x = request.POST["x"]
            new_data = ''
            input = open('/home/app/web/input/input1',
'r', encoding='utf-8')
            input_f = File(input)
            old_data = input_f.readlines()
            for l in old_data:
                if l.find('//Параметр X\n') != -1:
                    l = 'x=' + x + '//Параметр X\n'
                    new_data = new_data + str(l)
            output = open('/home/app/web/input/input1',
'w', encoding='utf-8')
            output_f = File(output)
            output_f.write(new_data)
            input_f.close()
            input.close()
            output_f.close()
            output.close()

        if request.POST["y"]:
            y = request.POST["y"]
            new_data = ''
            input = open('/home/app/web/input/input1',
'r', encoding='utf-8')
            input_f = File(input)
            old_data = input_f.readlines()
            for l in old_data:
                if l.find('//Параметр Y\n') != -1:
                    l = 'y=' + y + '//Параметр Y\n'
                    new_data = new_data + str(l)
```

```

        output = open('/home/app/web/input/input1',
'w', encoding='utf-8')
        output_f = File(output)
        output_f.write(new_data)
        input_f.close()
        input.close()
        output_f.close()
        output.close()

    if "box1" in request.POST:
        box1 = '1'
    else:
        box1= '0'
        new_data = ''
        input = open('/home/app/web/input/input1', 'r',
encoding='utf-8')
        input_f = File(input)
        old_data = input_f.readlines()
        for l in old_data:
            if l.find('//Флажок 1\n') != -1:
                l = 'box1=[' + box1 + ']{0|1}//Флажок
1\n'

                new_data = new_data + str(l)
        output = open('/home/app/web/input/input1', 'w',
encoding='utf-8')
        output_f = File(output)
        output_f.write(new_data)
        input_f.close()
        input.close()
        output_f.close()
        output.close()

    if "box2" in request.POST:
        box2 = '1'
    else:
        box2= '0'
        new_data = ''
        input = open('/home/app/web/input/input1', 'r',
encoding='utf-8')
        input_f = File(input)
        old_data = input_f.readlines()
        for l in old_data:
            if l.find('//Флажок 2\n') != -1:
                l = 'box2=[' + box2 + ']{0|1}//Флажок
2\n'

```

```

        new_data = new_data + str(l)
        output = open('/home/app/web/input/input1', 'w',
encoding='utf-8')
        output_f = File(output)
        output_f.write(new_data)
        input_f.close()
        input.close()
        output_f.close()
        output.close()

        if "box3" in request.POST:
            box3 = '1'
        else:
            box3= '0'
            new_data = ''
            input = open('/home/app/web/input/input1', 'r',
encoding='utf-8')
            input_f = File(input)
            old_data = input_f.readlines()
            for l in old_data:
                if l.find('//Флажок 3\n') != -1:
                    l = 'box3=[' + box3 + ']{0|1}//Флажок
3\n'
                    new_data = new_data + str(l)
            output = open('/home/app/web/input/input1', 'w',
encoding='utf-8')
            output_f = File(output)
            output_f.write(new_data)
            input_f.close()
            input.close()
            output_f.close()
            output.close()

        if "ParametersFile" in request.FILES:
            ParametersFile =
request.FILES["ParametersFile"]
            fs = FileSystemStorage()
            filename = fs.save(ParametersFile.name,
ParametersFile)
            file_url = fs.url(filename)

        return render(request, "input1.html")
    return render(request, "input1.html")

```



```

def input2(request):
    if request.method == "POST":
        if request.POST["X"]:
            X = request.POST["X"]
            new_data = ''
            input = open('/home/app/web/input/input2',
'r', encoding='utf-8')
            input_f = File(input)
            old_data = input_f.readlines()
            for l in old_data:
                if l.find('//Параметр X\n') != -1:
                    l = 'X=' + X + '//Параметр X\n'
                    new_data = new_data + str(l)
            output = open('/home/app/web/input/input2',
'w', encoding='utf-8')
            output_f = File(output)
            output_f.write(new_data)
            input_f.close()
            input.close()
            output_f.close()
            output.close()

        if request.POST["Y"]:
            Y = request.POST["Y"]
            new_data = ''
            input = open('/home/app/web/input/input2',
'r', encoding='utf-8')
            input_f = File(input)
            old_data = input_f.readlines()
            for l in old_data:
                if l.find('//Параметр Y\n') != -1:
                    l = 'Y=' + Y + '//Параметр Y\n'
                    new_data = new_data + str(l)
            output = open('/home/app/web/input/input2',
'w', encoding='utf-8')
            output_f = File(output)
            output_f.write(new_data)
            input_f.close()
            input.close()
            output_f.close()
            output.close()

        if "box1" in request.POST:
            box1 = '1'
        else:

```

```

        box1= '0'
        new_data = ''
        input = open('/home/app/web/input/input2', 'r',
encoding='utf-8')
        input_f = File(input)
        old_data = input_f.readlines()
        for l in old_data:
            if l.find('//Флажок 1\n') != -1:
                l = 'box1=[' + box1 + ']{0|1}//Флажок
1\n'
                new_data = new_data + str(l)
        output = open('/home/app/web/input/input2', 'w',
encoding='utf-8')
        output_f = File(output)
        output_f.write(new_data)
        input_f.close()
        input.close()
        output_f.close()
        output.close()

        if "box2" in request.POST:
            box2 = '1'
        else:
            box2= '0'
            new_data = ''
            input = open('/home/app/web/input/input2', 'r',
encoding='utf-8')
            input_f = File(input)
            old_data = input_f.readlines()
            for l in old_data:
                if l.find('//Флажок 2\n') != -1:
                    l = 'box2=[' + box2 + ']{0|1}//Флажок
2\n'
                    new_data = new_data + str(l)
            output = open('/home/app/web/input/input2', 'w',
encoding='utf-8')
            output_f = File(output)
            output_f.write(new_data)
            input_f.close()
            input.close()
            output_f.close()
            output.close()

        if "box3" in request.POST:
            box3 = '1'

```

```

else:
    box3= '0'
    new_data = ''
    input = open('/home/app/web/input/input2', 'r',
encoding='utf-8')
    input_f = File(input)
    old_data = input_f.readlines()
    for l in old_data:
        if l.find('//Флажок 3\n') != -1:
            l = 'box3=[' + box3 + ']{0|1}//Флажок
3\n'
            new_data = new_data + str(l)
    output = open('/home/app/web/input/input2', 'w',
encoding='utf-8')
    output_f = File(output)
    output_f.write(new_data)
    input_f.close()
    input.close()
    output_f.close()
    output.close()

if "ffile" in request.FILES:
    ffile = request.FILES["ffile"]
    fs = FileSystemStorage()
    filename = fs.save(ffile.name, ffile)
    file_url = fs.url(filename)

if "box4" in request.POST:
    box4 = '1'
else:
    box4= '0'
    new_data = ''
    input = open('/home/app/web/input/input2', 'r',
encoding='utf-8')
    input_f = File(input)
    old_data = input_f.readlines()
    for l in old_data:
        if l.find('//Флажок\n') != -1:
            l = 'box4=[' + box4 + ']{0|1}//Флажок\n'
            new_data = new_data + str(l)
    output = open('/home/app/web/input/input2', 'w',
encoding='utf-8')
    output_f = File(output)
    output_f.write(new_data)
    input_f.close()

```

```

input.close()
output_f.close()
output.close()

if request.POST["z"]:
    z = request.POST["z"]
    new_data = ''
    input = open('/home/app/web/input/input2',
'r', encoding='utf-8')
    input_f = File(input)
    old_data = input_f.readlines()
    for l in old_data:
        if l.find('//Параметр Z\n') != -1:
            l = 'z=' + z + '//Параметр Z\n'
            new_data = new_data + str(l)
    output = open('/home/app/web/input/input2',
'w', encoding='utf-8')
    output_f = File(output)
    output_f.write(new_data)
    input_f.close()
    input.close()
    output_f.close()
    output.close()

    return render(request, "input2.html")
return render(request, "input2.html")

def input3(request):
    if request.method == "POST":
        if request.POST["x"]:
            x = request.POST["x"]
            new_data = ''
            input = open('/home/app/web/input/input3',
'r', encoding='utf-8')
            input_f = File(input)
            old_data = input_f.readlines()
            for l in old_data:
                if l.find('//Параметр X\n') != -1:
                    l = 'x=' + x + '//Параметр X\n'
                    new_data = new_data + str(l)
            output = open('/home/app/web/input/input3',
'w', encoding='utf-8')
            output_f = File(output)
            output_f.write(new_data)

```

```

        input_f.close()
        input.close()
        output_f.close()
        output.close()

    if request.POST["y"]:
        y = request.POST["y"]
        new_data = ''
        input = open('/home/app/web/input/input3',
'r', encoding='utf-8')
        input_f = File(input)
        old_data = input_f.readlines()
        for l in old_data:
            if l.find('//Параметр Y\n') != -1:
                l = 'y=' + y + '//Параметр Y\n'
                new_data = new_data + str(l)
        output = open('/home/app/web/input/input3',
'w', encoding='utf-8')
        output_f = File(output)
        output_f.write(new_data)
        input_f.close()
        input.close()
        output_f.close()
        output.close()

    if request.POST["z"]:
        z = request.POST["z"]
        new_data = ''
        input = open('/home/app/web/input/input3',
'r', encoding='utf-8')
        input_f = File(input)
        old_data = input_f.readlines()
        for l in old_data:
            if l.find('//Параметр Z\n') != -1:
                l = 'z=' + z + '//Параметр Z\n'
                new_data = new_data + str(l)
        output = open('/home/app/web/input/input3',
'w', encoding='utf-8')
        output_f = File(output)
        output_f.write(new_data)
        input_f.close()
        input.close()
        output_f.close()
        output.close()

```

```

        if request.POST["q"]:
            q = request.POST["q"]
            new_data = ''
            input = open('/home/app/web/input/input3',
'r', encoding='utf-8')
            input_f = File(input)
            old_data = input_f.readlines()
            for l in old_data:
                if l.find('//Параметр Q\n') != -1:
                    l = 'q=' + q + '//Параметр Q\n'
                    new_data = new_data + str(l)
            output = open('/home/app/web/input/input3',
'w', encoding='utf-8')
            output_f = File(output)
            output_f.write(new_data)
            input_f.close()
            input.close()
            output_f.close()
            output.close()

    return render(request, "input3.html")
return render(request, "input3.html")

```

После запуска web-приложения, открывается главное меню (рис. 10). На данной странице можно выбрать один из трех ранее описанных вариантов.

## Здравствуй! Выберите один из предложенных вариантов.

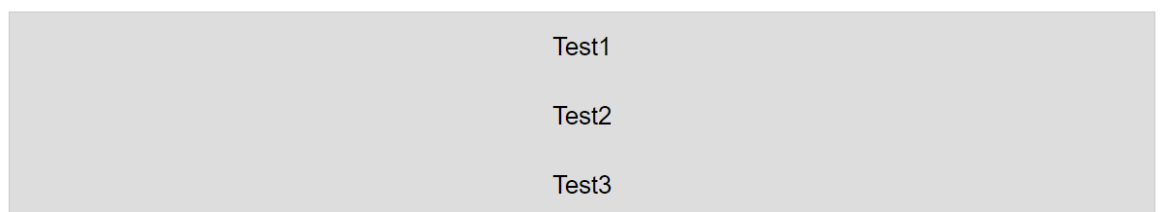


Рис. 10. Главное меню

При выборе первого варианта Test1 происходит переход по ссылке <http://195.19.40.68:8084/F1/> и открывается страница показанная на рис. 11, в которой уже присутствуют некоторые значения по умолчанию. Можно переключаться между вкладками данной страницы, что показано на рис. 12 (в данном случае вкладки 2).

Вкладка 1	Вкладка 2
-----------	-----------

**Параметр X**

**Параметр Y**

☐ Флажок 1

☒ Флажок 2

Отправить

Назад

Рис. 11. Страница после перехода по первой ссылке

Вкладка 1	Вкладка 2
-----------	-----------

**Параметр Q**

☐ Флажок 3

**Выберите требуемый файл**

Выберите файл

Файл не выбран

Московский государственный технический университет им. Н. Э. Баумана - фкпфроссийский национальный исследовательский университет, научный центр, особо ценный объект культурного наследия народов России.

[Дополнительная информация](#)

Отправить

Назад

Рис. 12. Демонстрация перехода на вторую вкладку

Были введены значения в поля ввода, нажаты некоторые флажки, и выбран файл (рисунки 13 и 14).

Вкладка 1	Вкладка 2
<p><b>Параметр X</b></p> <input type="text" value="3600"/>	
<p><b>Параметр Y</b></p> <input type="text" value="3070"/>	
<p><input checked="" type="checkbox"/> Флажок 1</p> <p><input type="checkbox"/> Флажок 2</p>	
<p>Отправить</p>	
<p>Назад</p>	

Рис. 13. Первая вкладка страницы с новыми значениями

Вкладка 1	Вкладка 2
<p><b>Параметр Q</b></p> <input type="text" value="16"/>	
<p><input checked="" type="checkbox"/> Флажок 3</p>	
<p><b>Выберите требуемый файл</b></p> <p>Выберите файл <input type="text" value="bmstu.png"/></p>	
<p>Московский государственный технический университет им. Н. Э. Баумана - фкпфроссийский национальный исследовательский университет, научный центр, особо ценный объект культурного наследия народов России.</p> <p><a href="#">Дополнительная информация</a></p>	
<p>Отправить</p>	
<p>Назад</p>	

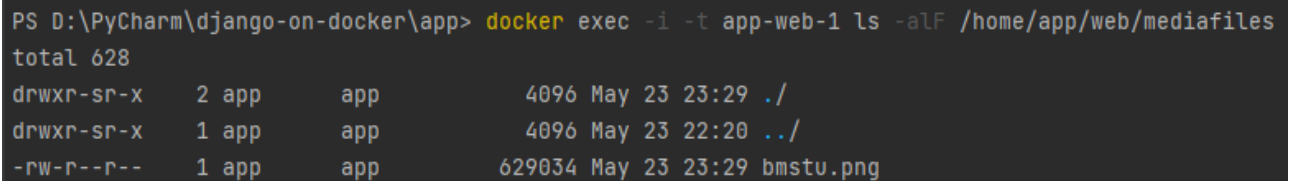
Рис. 14. Вторая вкладка страницы с новыми значениями

После нажатия кнопки «Отправить» значения были записаны в исходный aINI файл, что можно увидеть на листинге 36, а выбранный файл был сохранен в папку mediafiles в докере (рис. 15), откуда его можно загрузить в папку, находящуюся у хоста с помощью команды “ `docker cp app-web-1:/home/app/web/input/bmstu.png D:\PyCharm\django-on-docker`”.



### Листинг. 36. Скорректированный файл с aINI данными страницы

```
[sec1]//Вкладка 1
x=3600//Параметр X
y=3070//Параметр Y
box1=[1]{0|1}//Флажок 1
box2=[0]{0|1}//Флажок 2
[sec2]//Вкладка 2
q=16//Параметр Q
box3=[1]{0|1}//Флажок 3
ParametersFile=[file]//Выберите требуемый файл
//Московский государственный технический университет им.
Н. Э. Баумана - фкпфроссийский национальный
исследовательский университет, научный центр, особо
ценный объект культурного наследия народов России.
[https://bmstu.ru/about]//Дополнительная информация
```



```
PS D:\PyCharm\django-on-docker\app> docker exec -i -t app-web-1 ls -alF /home/app/web/mediafiles
total 628
drwxr-sr-x  2 app    app      4096 May 23 23:29 ./
drwxr-sr-x  1 app    app      4096 May 23 22:20 ../
-rw-r--r--  1 app    app    629034 May 23 23:29 bmstu.png
```

Рис. 15. Демонстрация присутствия в докере сохранённого файла

## ЗАКЛЮЧЕНИЕ

- Были рассмотрены существующие подходы разработки GUI;
- было теоретическое и практическое знакомство с Django и Docker, после чего было разработано web-приложение;
- была разработана программа для преобразования данных формата aINI в HTML-код;
- web-приложение было запущено на сервере с использованием интерфейса, сгенерированного разработанной программой;
- автоматизированное построение GUI на основе данных в формате с простым синтаксисом (например, aINI) позволяет быстро создавать несколько графических интерфейсов пользователям без знаний программирования.
- разработанное программное решение удобно тем, что формат ввода и вывода стандартизированы, что позволит пользователю использовать полученные данные для повторной генерции.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Соколов А.П., Описание формата данных aINI (advanced INI). 2020
- [2] Лукьянов Д.В., Разработка графического пользовательского интерфейса // Новые информационные технологии в автоматизированных системах. 2012
- [3] Санковский Ю.Е., Метод построения оконного интерфейса пользователя на основе моделирования пользовательских целей. 1998
- [4] Казаков Г.В., Корянов В.В., Чемирисов В.В., Уваров А.В. Методический подход к созданию универсального пользовательского интерфейса. 2020.
- [5] Юркин В.А., Сараджишвили С.Э. Построение пользовательского интерфейса с использованием интерактивного машинного обучения. 2020.
- [6] MDN Web Docs. // runebook.dev — URL: <https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction>. (Дата обращения 24.10.2022).
- [7] Eternalhost. // runebook.dev — URL: [https://eternalhost.net/blog/razrabotka/chto-takoe-docker?utm\\_source=google.com&utm\\_medium=organic&utm\\_campaign=google.com&utm\\_referrer=google.com](https://eternalhost.net/blog/razrabotka/chto-takoe-docker?utm_source=google.com&utm_medium=organic&utm_campaign=google.com&utm_referrer=google.com). (Дата обращения 24.10.2022).
- [8] Xgu.ru. // <http://xgu.ru/wiki/pyarsing#:~:text=%5B%D%20Pyarsing%20—%20модуль%20синтаксического,достаточно%20большое%20количество%20синтаксических%20анализаторов>. (Дата обращения 28.02.2023)
- [9] Python Documentation // <https://docs.python.org/3/tutorial/errors.html> (Дата обращения 28.02.2023)