



МИНОБРАЗОВАНИЯ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных
технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2
по дисциплине

«Математическое обеспечение информационных технологий»

Тема: «Работа с Bash Scripting»

Выполнил студент группы ИТ-12-23

Принял

Иванов И.В.

Генеральный директор

Практическая работа выполнена

«18» 03 2025 г.

Иванов И.В.
(подпись студента)

«Зачтено»

«18» 03 2025 г.

Иванов И.В.
(подпись руководителя)

Москва 202__

Оглавление

1. Введение.....	3
1.1 Цели работы.....	3
2. Ход работы. Базовые Bash скрипты	4
2.1 Задание 1	4
2.2 Задание 2	5
2.3 Задание 3	6
2.4 Задание 4	6
2.5 Задание 5	7
2.6 Задание 6	8
3. Ход работы. Развертка и запуск проекта при помощи Bash Script .	10
3.1 Определение зависимостей проекта	10
3.2 Создание виртуального окружения	11
3.3 Написание скрипта запуска приложения на новой системе	13
4. Выводы	15
4.1 Выводы	15

1. ВВЕДЕНИЕ

1.1 Цели работы

Данная работа посвящена основам написания Bash-скриптов. Bash-скрипты являются мощным инструментом автоматизации в Unix-подобных системах, позволяя объединять последовательности команд в единые исполняемые файлы для эффективного решения разнообразных задач. В основе любого скрипта лежит шебанг-строка `#!/bin/bash`, которая указывает системе на необходимость использования интерпретатора Bash для обработки содержащихся в файле инструкций. После указания интерпретатора следует основной код, состоящий из набора команд, выполняемых последовательно сверху вниз.

Язык Bash предоставляет разработчикам богатый набор возможностей, включая работу с переменными для хранения данных, использование арифметических операций для вычислений, подстановку результатов выполнения команд, а также сложные управляющие конструкции.

Освоение базовых элементов, таких как объявление переменных, выполнение системных команд, организация условий и циклов, а также проверка состояния файловой системы, формирует необходимую основу для дальнейшего углубленного изучения Bash-программирования.

2. ХОД РАБОТЫ. БАЗОВЫЕ BASH СКРИПТЫ

2.1 Задание 1

Напишите сценарий, который выводит дату, время, список зарегистрировавшихся пользователей, и uptime системы и сохраняет эту информацию в файл. Программная реализация данного задания представлена на рисунке 1, а результат скрипта на рисунке 2.

```
ilal@VeX MINGW64 ~/TRPP/blocknote-master
$ cat system_info.sh
#!/bin/bash

file="system_info.txt"

# Очищаем файл или создаем новый
> "$file"

# 1. Дата и время
echo "===== Системная информация =====" >> "$file"
echo "- Дата и время: $(date "+%Y-%m-%d %H:%M:%S")" >> "$file"

# 2. Пользователи (для Windows)
echo "- Пользователи системы:" >> "$file"
whoami >> "$file"
echo "" >> "$file"

# 3. Uptime (альтернатива для Windows)
if command -v wmic &> /dev/null; then
    last_boot=$(wmic os get lastbootuptime | grep -oP '\d{14}')
    if [ -n "$last_boot" ]; then
        boot_time=$(echo "$last_boot" | awk '{print substr($0,1,4)"-"substr($0,5,2)"-"substr($0,7,2)" "s
substr($0,9,2)" ":"substr($0,11,2)" ":"substr($0,13,2)}')
        boot_sec=$(date -d "$boot_time" +%s)
        now_sec=$(date +%s)
        uptime_sec=$((now_sec - boot_sec))

        days=$((uptime_sec/86400))
        hours=$(( (uptime_sec%86400)/3600 ))
        mins=$(( (uptime_sec%3600)/60 ))

        echo "- Время работы системы:" >> "$file"
        printf " %d дней %02d:%02d\n" "$days" "$hours" "$mins" >> "$file"
    else
        echo "- Время работы: не удалось определить" >> "$file"
    fi
else
    echo "- Время работы: команда wmic недоступна" >> "$file"
fi

echo "Информация сохранена в файл: $file"
```

Рисунок 1 – Программная реализация задания 1

```

ilalb@VeX MINGW64 ~/TRPP/blocknote-master
$ cat system_info.txt
===== Системная информация =====
- Дата и время: 2025-04-01 17:49:23
- Пользователи системы:
ilalb

- Время работы системы:
  3 дней 17:54

```

Рисунок 2 – Дата, время, список пользователей и uptime

2.2 Задание 2

Напишите сценарий, который выводит содержимое любого каталога или сообщение о том, что его не существует. Программная реализация скрипта представлена на рисунке 3, а тестирование на рисунке 4.

```

ilalb@VeX MINGW64 ~/TRPP/blocknote-master
$ touch list_dir.sh; nano list_dir.sh; cat list_dir.sh; ./list_dir.sh
#!/bin/bash

echo "Enter directory path:"
read dir_path

if [ -d "$dir_path" ]; then
    echo "Contents of $dir_path:"
    ls -l "$dir_path"
else
    echo "Directory $dir_path does not exist."
fi

```

Рисунок 3 – Программная реализация задания 2

```

ilalb@VeX MINGW64 ~/TRPP/blocknote-master
$ ./list_dir.sh
Enter directory path:
test
Contents of test:
total 0

ilalb@VeX MINGW64 ~/TRPP/blocknote-master
$ cd test

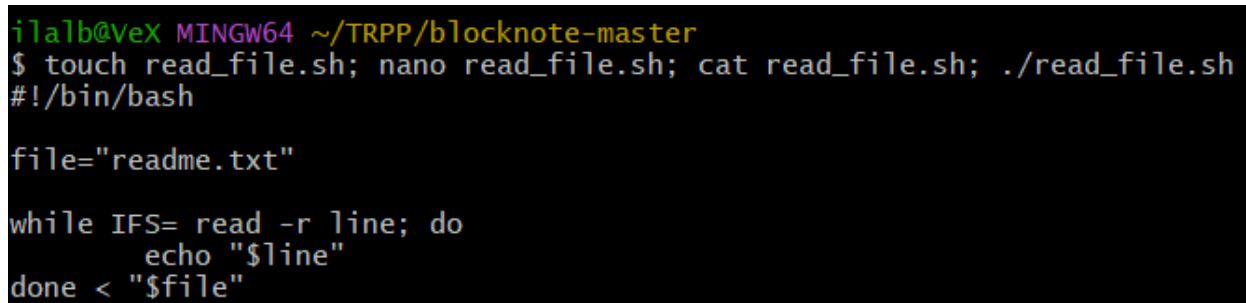
ilalb@VeX MINGW64 ~/TRPP/blocknote-master/test
$ ls

```

Рисунок 4 – Тестирование скрипта для задания 2

2.3 Задание 3

Напишите сценарий, который с помощью цикла прочитает файл и выведет его содержимое. Программная реализация скрипта и результат представлены на рисунках 5-6 соответственно.

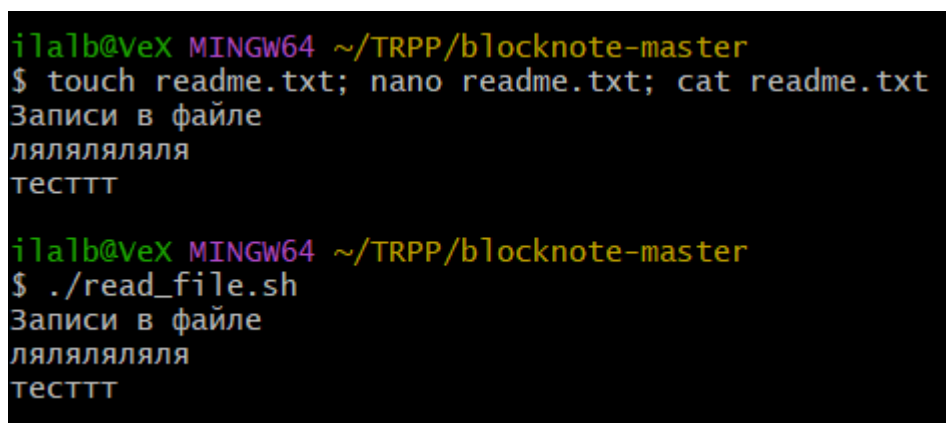


```
ila1b@VeX MINGW64 ~/TRPP/blocknote-master
$ touch read_file.sh; nano read_file.sh; cat read_file.sh; ./read_file.sh
#!/bin/bash

file="readme.txt"

while IFS= read -r line; do
    echo "$line"
done < "$file"
```

Рисунок 5 – Программная реализация задания 3



```
ila1b@VeX MINGW64 ~/TRPP/blocknote-master
$ touch readme.txt; nano readme.txt; cat readme.txt
Записи в файле
ляляляляля
тесттт

ila1b@VeX MINGW64 ~/TRPP/blocknote-master
$ ./read_file.sh
Записи в файле
ляляляляля
тесттт
```

Рисунок 6 – Построчное чтение файла

2.4 Задание 4

Напишите сценарий, который с помощью цикла выведет список файлов и директорий из текущего каталога, укажет, что есть файл, а что директория. Программная реализация скрипта и результат представлены на рисунках 7-8 соответственно.

```

ila1b@VeX MINGW64 ~/TRPP/blocknote-master
$ touch list_files_types.sh; nano list_files_types.sh; cat list_files_types.sh; ./list_files_types.sh
#!/bin/bash

for item in *; do
    if [ -f "$item" ]; then
        echo "[File] $item"
    elif [ -d "$item" ]; then
        echo "[Directory] $item"
    fi
done

```

Рисунок 7 – Программная реализация задания 4

```

[Directory] appengine
[Directory] apps
[File] list_dir.sh
[File] list_files_types.sh
[File] manage.py
[File] read_file.sh
[File] readme.txt
[File] requirements.txt
[File] run_project.sh
[File] start.sh
[Directory] static
[File] system_info.sh
[File] system_info.txt
[Directory] templates
[Directory] test

ila1b@VeX MINGW64 ~/TRPP/blocknote-master
$ ls
appengine/  list_files_types.sh*  readme.txt  start.sh*  system_info.txt
apps/      manage.py*           requirements.txt  static/    templates/
list_dir.sh*  read_file.sh*       run_project.sh*  system_info.sh*  test/

```

Рисунок 8 – Содержимое каталога

2.5 Задание 5

Напишите сценарий, который подсчитает объем диска, занимаемого директорией. В качестве директории можно выбрать любую директорию в системе. Программная реализация скрипта и результат представлены на рисунках 9-10 соответственно.

```

ilalb@VeX MINGW64 ~/TRPP/blocknote-master
$ touch dir_size.sh; nano dir_size.sh; cat dir_size.sh;
#!/bin/bash

echo "Enter directory path:"
read dir_path

if [ -d "$dir_path" ]; then
    du -sh "$dir_path"
else
    echo "Directory $dir_path does not exist."
fi

```

Рисунок 9 – Программная реализация задания 5

```

ilalb@VeX MINGW64 ~/TRPP/blocknote-master
$ ./dir_size.sh
Enter directory path:
apps
86K      apps

```

Рисунок 10 – Объем диска, занимаемый текущей директорией

2.6 Задание 6

Напишите сценарий, который выведет список всех исполняемых файлов в директории, для которых у текущего пользователя есть права на исполнение. Программная реализация скрипта и результат представлены на рисунках 11-12 соответственно.

```

$ touch ex_files.sh; nano ex_files.sh; cat ex_files.sh; ./ex_files.sh
#!/bin/bash

echo "Исполняемые файлы с правами на исполнение:"
find "$(pwd)" -type f -executable -print

```

Рисунок 11 – Программная реализация задания 6


```
Исполняемые файлы с правами на исполнение:  
/c/Users/ila1b/TRPP/blocknote-master/dir_size.sh  
/c/Users/ila1b/TRPP/blocknote-master/ex_files.sh  
/c/Users/ila1b/TRPP/blocknote-master/list_dir.sh  
/c/Users/ila1b/TRPP/blocknote-master/list_files_types.sh  
/c/Users/ila1b/TRPP/blocknote-master/manage.py  
/c/Users/ila1b/TRPP/blocknote-master/read_file.sh  
/c/Users/ila1b/TRPP/blocknote-master/run_project.sh  
/c/Users/ila1b/TRPP/blocknote-master/start.sh  
/c/Users/ila1b/TRPP/blocknote-master/system_info.sh
```

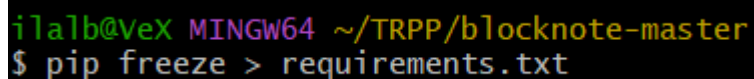
Рисунок 12 – Все исполняемые файлы в текущей директории

3. ХОД РАБОТЫ. РАЗВЕРТКА И ЗАПУСК ПРОЕКТА ПРИ ПОМОЩИ BASH SCRIPT

3.1 Определение зависимостей проекта

Файл «requirements.txt» играет важную роль в управлении зависимостями Python-проекта. Он содержит список всех необходимых библиотек и их версий, что позволяет быстро воспроизвести рабочую среду на любом компьютере или сервере. Это обеспечивает воспроизводимость, автоматизацию установки зависимостей и контроль версий пакетов, предотвращая конфликты и ошибки.

В ходе данной работы необходимо составить файл зависимостей для [следующего проекта](#) (доступен по ссылке). Генерация файла и его содержимого представлена на рисунках 13-14.



```
ilalb@VeX MINGW64 ~/TRPP/blocknote-master  
$ pip freeze > requirements.txt
```

Рисунок 13 – Автоматическая генерация файла requirements.txt

```
ilalb@VeX MINGW64 ~/TRPP/blocknote-master
$ cat requirements.txt
asttokens==3.0.0
colorama==0.4.6
comm==0.2.2
debugpy==1.8.12
decorator==5.1.1
executing==2.2.0
ipykernel==6.29.5
ipython==8.32.0
jedi==0.19.2
jupyter_client==8.6.3
jupyter_core==5.7.2
matplotlib-inline==0.1.7
nest-asyncio==1.6.0
packaging==24.2
parso==0.8.4
platformdirs==4.3.6
prompt_toolkit==3.0.50
psutil==7.0.0
pure_eval==0.2.3
Pygments==2.19.1
python-dateutil==2.9.0.post0
pywin32==308
pyzmq==26.2.1
six==1.17.0
stack-data==0.6.3
tornado==6.4.2
traitlets==5.14.3
wcwidth==0.2.13
```

Рисунок 14 – Список зависимостей для проекта

3.2 Создание виртуального окружения

Необходимо на основании составленного в прошлом шаге списка команд написать скрипт скачивания указанного в прошлом шаге проекта с последующим созданием виртуального окружения и настройкой его под проект, то есть установкой всех необходимых библиотек. Скрипт для создания виртуального окружения представлен на рисунке 15, процесс работы скрипта виден на рисунках 16-17.

```

$ cat download_script.sh
#!/bin/bash
ARCHIVE_NAME="blocknote-master.tar.gz"
PROJECT_FOLDER="blocknote-master"
PROJECT_URL="https://www.dropbox.com/s/ija7ax3sj6ysb0p/blocknote-master.tar.gz"

echo "===== Скачивание проекта... ====="
curl -L -o "$ARCHIVE_NAME" "$PROJECT_URL"

echo "Распаковка архива..."
tar -xzf "$ARCHIVE_NAME"

cd "$PROJECT_FOLDER" || exit
echo "PWD: $(pwd)"

echo "===== Создание виртуального окружения... ====="
python -m venv venv

echo "Активация виртуального окружения..."
if [ -f "venv/Scripts/activate" ]; then
    source venv/Scripts/activate
else
    source venv/bin/activate
fi

echo "Проверяем, запущено ли виртуальное окружение..."
if [[ -z "$VIRTUAL_ENV" ]]; then
    echo "ERROR: Виртуальное окружение не найдено. Пожалуйста, активируйте его перед запуском скрипта."
    exit 1
else
    echo "Виртуальное окружение активно"
fi

echo "===== Генерация полного файла requirements.txt... ====="
echo "Текущий VIRTUAL_ENV: $VIRTUAL_ENV"

echo "===== Установка зависимостей... ====="
if [ -f "requirements.txt" ]; then
    echo "Найден requirements.txt, устанавливаем зависимости..."
    if [ -s "requirements.txt" ]; then
        pip install -r requirements.txt
    else
        echo "ERROR: файл requirements.txt пустой - зависимости не обнаружены"
    fi
else
    echo "ERROR: Не найден requirements.txt"
fi

echo "===== Установка проекта завершена ====="

```

Рисунок 15 – Скачивание проекта и создание виртуального окружения

```

$ ./download_script.sh
===== Скачивание проекта... =====
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left     Speed
100  131  100  131    0    0   400      0 --:--:-- --:--:-- --:--:--   401
100  17  100  17    0    0   20      0 --:--:-- --:--:-- --:--:--   20
100  470    0  470    0    0   285      0 --:--:-- 0:00:01 --:--:--    0
100 12.0M 100 12.0M    0    0 3443k      0 0:00:03 0:00:03 --:--:-- 8877k
Распаковка архива...
blocknote-master/
blocknote-master/..DS_Store
blocknote-master/.DS_Store
blocknote-master/appengine/
blocknote-master/..gitignore

```

Рисунок 16 – Скачивание проекта и создание виртуального окружения. Часть 1

```

Using cached pure_eval-0.2.3-py3-none-any.whl (11 kB)
Using cached pygments-2.19.1-py3-none-any.whl (1.2 MB)
Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Using cached pywin32-308-cp313-cp313-win_amd64.whl (6.5 MB)
Using cached pyzmq-26.2.1-cp313-cp313-win_amd64.whl (643 kB)
Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Using cached stack_data-0.6.3-py3-none-any.whl (24 kB)
Using cached tornado-6.4.2-cp38-abi3-win_amd64.whl (438 kB)
Using cached traitlets-5.14.3-py3-none-any.whl (85 kB)
Using cached wcwidth-0.2.13-py2.py3-none-any.whl (34 kB)
Installing collected packages: wcwidth, pywin32, pure_eval, traitlets, tor
a, asttokens, stack_data, python_dateutil, matplotlib-inline, jupyter_core
Successfully installed Pygments-2.19.1 asttokens-3.0.0 colorama-0.4.6 comm
2 matplotlib-inline-0.1.7 nest-asyncio-1.6.0 packaging-24.2 parso-0.8.4 pl
k-data-0.6.3 tornado-6.4.2 traitlets-5.14.3 wcwidth-0.2.13

[notice] A new release of pip is available: 24.3.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
===== Установка проекта завершена =====

```

Рисунок 17 - Скачивание проекта и создание виртуального окружения. Часть 2

3.3 Написание скрипта запуска приложения на новой системе

Пришло время запустить проект, при этом воссоздав полученное на прошлом этапе виртуальное окружение со всеми зависимостями, после чего запустить код, представленный на рисунке 18.

```

blocknote-master > start-project.sh
1  python manage.py makemigrations
2  python manage.py migrate
3  python manage.py runserver

```

Рисунок 18 – Программная реализация запуска проекта

Проект настроен и готов к запуску (рис. 19-20).

```

System check identified some issues:

WARNINGS:
2: (ckeditor.w001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported anymore and which does have unfixed security issues, see for example https://ckeditor.com/ckeditor4/release/CKEditor-4.24.0-LTS . You should consider strongly switching to a different editor (maybe CKEditor 5 respectively django-ckeditor-5 after checking whether the CKEditor 5 license terms work for you) or switch to the non-free CKEditor 4 LTS package. See https://ckeditor.com/ckeditor-4-support/ for more on this. (Note! This notice has been added by the django-ckeditor developers and we are not affiliated with CKSource and were not involved in the licensing change, so please refrain from complaining to us. Thanks.)
main.Article: (models.w042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
      HINT: Configure the DEFAULT_AUTO_FIELD setting or the MainConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
main.GroupArticles: (models.w042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
      HINT: Configure the DEFAULT_AUTO_FIELD setting or the MainConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
todapp.Task: (models.w042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
      HINT: Configure the DEFAULT_AUTO_FIELD setting or the BaseConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.

No changes detected.

System check identified some issues:

WARNINGS:
2: (ckeditor.w001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported anymore and which does have unfixed security issues, see for example https://ckeditor.com/ckeditor4/release/CKEditor-4.24.0-LTS . You should consider strongly switching to a different editor (maybe CKEditor 5 respectively django-ckeditor-5 after checking whether the CKEditor 5 license terms work for you) or switch to the non-free CKEditor 4 LTS package. See https://ckeditor.com/ckeditor-4-support/ for more on this. (Note! This notice has been added by the django-ckeditor developers and we are not affiliated with CKSource and were not involved in the licensing change, so please refrain from complaining to us. Thanks.)
main.Article: (models.w042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
      HINT: Configure the DEFAULT_AUTO_FIELD setting or the MainConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
main.GroupArticles: (models.w042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
      HINT: Configure the DEFAULT_AUTO_FIELD setting or the MainConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
todapp.Task: (models.w042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
      HINT: Configure the DEFAULT_AUTO_FIELD setting or the BaseConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.

Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions, todapp
Running migrations:
  No migrations to apply.
Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:

WARNINGS:
2: (ckeditor.w001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported anymore and which does have unfixed security issues, see for example https://ckeditor.com/ckeditor4/release/CKEditor-4.24.0-LTS . You should consider strongly switching to a different editor (maybe CKEditor 5 respectively django-ckeditor-5 after checking whether the CKEditor 5 license terms work for you) or switch to the non-free CKEditor 4 LTS package. See https://ckeditor.com/ckeditor-4-support/ for more on this. (Note! This notice has been added by the django-ckeditor developers and we are not affiliated with CKSource and were not involved in the licensing change, so please refrain from complaining to us. Thanks.)
main.Article: (models.w042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
      HINT: Configure the DEFAULT_AUTO_FIELD setting or the MainConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
main.GroupArticles: (models.w042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
      HINT: Configure the DEFAULT_AUTO_FIELD setting or the MainConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
todapp.Task: (models.w042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
      HINT: Configure the DEFAULT_AUTO_FIELD setting or the BaseConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.

System check identified 4 issues (0 silenced).
March 10, 2025 - 18:47:17
Django version 5.1.7, using settings 'appengine.settings'
Starting the development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

```

Рисунок 19 – Запуск проекта

Please sign in

Email address

or [Sign Up](#)

Рисунок 20 – Запущенный проект

4. ВЫВОДЫ

4.1 Выводы

В рамках практической работы был детально проанализирован процесс разработки и применения Bash-скриптов в Unix-средах. Основное внимание уделялось фундаментальным аспектам, включая синтаксис шобанг-строки, работу с переменными, механизмы подстановки команд, выполнение арифметических операций, а также применение управляющих конструкций и циклических структур.

Особое значение в работе было уделено вопросам управления зависимостями в программных проектах. Процедура идентификации необходимых компонентов играет критическую роль в обеспечении стабильной работы приложений в различных средах исполнения. Формирование файла requirements.txt, содержащего исчерпывающий перечень требуемых библиотек, существенно упрощает процессы развёртывания и настройки рабочего окружения.

Практическое освоение Bash-скриптинга открывает значительные перспективы для системных администраторов и разработчиков, работающих в Unix-подобных средах. Полученные знания базовых принципов создания скриптов служат отправной точкой для разработки более сложных и функциональных решений.