Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий  
Кафедра «Информатика и информационные технологии»

Направление подготовки/ специальность: Информационные системы и технологии/ Автоматизированные системы обработки информации и управления

ОТЧЕТ

по проектной практике

Студент: Чернышёва Ольга Вадимовна Группа: 241-339

Место прохождения практики: Московский Политех, кафедра «информатика и информационные технологии»

Отчет принят с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики: Меньшикова Наталия Павловна

Москва 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

1. Общая информация о проекте:

* Название проекта
* Цели и задачи проекта

1. Общая характеристика деятельности организации *(заказчика проекта)*

* Наименование заказчика
* Организационная структура
* Описание деятельности

1. Описание задания по проектной практике
2. Описание достигнутых результатов по проектной практике

ЗАКЛЮЧЕНИЕ *(выводы о проделанной работе и оценка ценности выполненных задач для заказчика)*

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

ПРИЛОЖЕНИЯ *(при необходимости)*

1. ОБЩАЯ ИНФОРМАЦИЯ О ПРОЕКТЕ

- Название проекта: Группа игровой компьютерной индустрии «Семилучье»

- Цели и задачи проекта:

**Цель:**

Предоставить пользователям готовый продукт для удовольствия и развлечения.

**Задачи:**

1. **Разработка концепции игры** (Определить жанр, целевую аудиторию и основные механики игры.);
2. **Планирование проекта** (Составить план проекта с указанием сроков, этапов);
3. **Создание игрового мира и персонажей;**
4. **Дизайн уровней и механик:**

* Создать карты уровней и определить их последовательность
* Разработать препятствия, мини-игры и квесты
* Настроить сложность  и баланс между различными элементами игры

1. **Разработка и интеграция графики**:

* Выбрать игровой движок
* Реализовать основные механики игры (движение, взаимодействие, физика)
* Интегрировать спрайты и анимации в игру

1. **Создание пользовательского интерфейса (UI);**
2. **Написание сценария и диалогов;**
3. **Тестирование и оптимизация.**
4. ОБЩАЯ ХАРАКТЕРИСТИКА ДЕЯТЕЛЬНОСТИ ОРГАНИЗАЦИИ *(ЗАКАЗЧИКА ПРОЕКТА)*

*-* Наименование заказчика: Федеральное государственное автономное образовательное учреждение высшего образования «МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

1. ОПИСАНИЕ ЗАДАНИЯ ПО ПРОЕКТНОЙ ПРАКТИКЕ
2. **Базовая часть задания**
3. **Настройка Git и репозитория:**

* Создайте личный или групповой репозиторий на GitHub или GitVerse на основе представленного шаблона.
* Освойте базовые команды Git: клонирование, коммит, пуш и создание веток.
* Регулярно фиксируйте изменения с осмысленными сообщениями к коммитам.
* **Ожидаемое время:** 5 часов.

1. **Написание документов в Markdown:**

* Все материалы проекта (описание, журнал прогресса и др.) должны быть оформлены в формате Markdown.
* Изучите синтаксис Markdown и подготовьте необходимые документы.
* **Ожидаемое время:** 5 часов.

1. **Создание статического веб-сайта:**

* Вы можете использовать **только HTML и CSS** для создания сайта, если освоение более сложных инструментов представляется трудным. Это делает задание доступным для студентов с базовым уровнем подготовки.
* **Желательно** применять генераторы статических сайтов, такие как Hugo (рекомендуется), для упрощения процесса и получения дополнительных навыков.
* Создайте новый сайт об основном проекте по дисциплине «Проектная деятельность», выберите тему и добавьте контент. Оформление и наполнение сайта должны быть уникальными (не совпадать с работами других студентов) более, чем на 50%.
* Сайт должен включать:
* **Домашнюю страницу** с аннотацией проекта.
* **Страницу «О проекте»** с описанием проекта.
* **Страницу или раздел «Участники»** с описанием личного вклада каждого участника группы в проект по «Проектной деятельности».
* **Страницу или раздел «Журнал»** с минимум тремя постами (новостями, блоками) о прогрессе работы.
* **Страницу «Ресурсы»** со ссылками на полезные материалы (ссылки на организацию-партнёра, сайты и статьи, позволяющие лучше понять суть проекта).
* Оформите страницы сайта графическими материалами (фотографиями, схемами, диаграммами, иллюстрациями) и другой медиа информацией (видео).
* **Ожидаемое время:** изучение и настройка — 10–14 часов, дизайн и наполнение — 4–8 часов.

1. **Взаимодействие с организацией-партнёром:**

* Организуйте взаимодействие с партнёрской организацией (визит, онлайн-встреча или стажировка).
* Участвуйте в профильных мероприятиях по тематике проекта и профилю организации-партнёра (конференции, выставки, митапы, семинары, хакатоны и др.).
* **Уточнение:** Взаимодействие осуществляется через куратора проекта по проектной деятельности, закреплённого за вашим проектом, и ответственного по проектной практике, закреплённого за учебной группой.
* Напишите отчёт в формате Markdown с описанием опыта, полученных знаний и связи с проектом. Отчёт добавьте в репозиторий и на сайт.
* **Важно:** Стажировки и экскурсии в организации-партнёры будут приниматься к зачёту и учитываться при оценке, что мотивирует к активному участию.
* **Ожидаемое время:** взаимодействие — 4 часа, написание отчёта — 4 часа.

1. **Отчёт по практике**

* Составьте отчёт по проектной (учебной) практике на основании шаблона (структуры), размещённого в папке reports. Шаблон (структура) приведён в файле [practice\_report\_template.docx](https://github.com/mospol/practice-2025-1/blob/master/task/reports/practice_report_template.docx).
* Разместите отчёт в репозитории в папке reports с именем «Отчёт.docx» или «report.docx».
* Сформируйте PDF-версию отчёта и также разместите её в папке reports в репозитории.
* Загрузите оба файла отчёта (DOCX и PDF) в СДО (LMS) в курсе, который будет указан ответственным за проектную (учебную) практику.

1. **Вариативная часть задания**

По решению ответственного за проектную (учебную) практику студентам назначается одно из следующих вариативных заданий. Студенты могут направить ответственному свои пожелания по распределению.

#### Практическая реализация технологии

* Выполните все задачи базовой части.
* Для достижения объёма в 72 часа выберите один из следующих проектов:

1. Выберите любую технологию (тематику) из списка, представленного в репозитории [codecrafters-io/build-your-own-x](https://github.com/codecrafters-io/build-your-own-x). По согласованию с ответственными за практику можно использовать другой источник проектов.
2. Согласуйте внутри команды выбранную тему. Выберите стек технологий (подсказки также есть в репозитории).
3. Проведите исследование: изучите, как создать выбранную технологию с нуля, воспроизведите практическую часть.
4. Создайте подробное описание в формате Markdown, включающее:

* Последовательность действий по исследованию предметной области и созданию технологии.
* Напишите техническое руководство по созданию этой технологии, ориентированное на начинающих.
* Включите в руководство пошаговые инструкции и примеры кода
* Иллюстрации (картинки, диаграммы, схемы) в количестве от 3 до 10 штук, вставленные в текст для наглядности.
* Поместите результаты исследования и руководства в общий Git-репозиторий.

1. Создайте техническое руководство или туториал по созданию проекта на выбранную тему. Для визуализации архитектуры, процессов и прочего используйте разные типы диаграмм UML, схемы, графики, таблицы.
2. Сделайте модификацию проекта согласно полученным знаниям и навыкам в течение года (творческий пункт, самостоятельно выбираете в какой части модифицировать). Описать в технической документации модификации.
3. Задокументируйте проект в репозитории в формате Markdown и представьте его на сайте в формате HTML.
4. Подготовить финальный отчет (в хронологической последовательности опишите этапы работы, отдельно должны быть представлены индивидуальные планы каждого участника).
5. ОПИСАНИЕ ДОСТИГНУТЫХ РЕЗУЛЬТАТОВ ПО ПРОЕКТНОЙ ПРАКТИКЕ

**Базовая часть**

**Проект: Настройка Git, создание статического сайта и взаимодействие с партнёрами**

## **Хронология выполнения задач**

### **1. Настройка Git и репозитория** (Ожидаемое время: 5 часов)

#### **Выполненные действия:**

* Создан репозиторий на **GitHub** на основе предоставленного шаблона.
* Изучены базовые команды Git:
  + git clone – клонирование репозитория,
  + git commit -m "сообщение" – фиксация изменений,
  + git push – отправка изменений на сервер,
  + git branch / git checkout – работа с ветками.
* Регулярно выполнялись коммиты с осмысленными сообщениями (например: "Добавлен README.md", "Исправлена структура проекта").

### ****2. Создание статического веб-сайта**** (Ожидаемое время: 14–22 часа)

#### ****Этапы работы:****

1. **Выбор инструмента:** Решено использовать **Hugo** (по рекомендации).
2. **Установка и настройка:**
   * Создан новый сайт через hugo new site myproject.
   * Выбрана и настроена тема (например, **Ananke**).
3. **Структура сайта:**
   * Главная страница (index.md) – аннотация проекта.
   * about.md – описание проекта.
   * team.md – вклад участников.
   * journal – журнал прогресса (минимум 3 поста).
   * resources.md – полезные ссылки.
4. **Дизайн и контент:**
   * Добавлены изображения, схемы.

### ****3. Взаимодействие с организацией-партнёром****

#### ****Выполненные действия:****

* Посещены рекомендованные мероприятия
* По каждому из мероприятий написан отчёт

## ****Итоги****

* Репозиторий настроен, сайт запущен.
* Документация оформлена в Markdown.
* Составлены отчеты по мероприятиям.

**Вариативная часть**

Финальный отчет по разработке Telegram-бота «Гороскоп»

\*\*Хронология работы над проектом\*\*

Этап 1: Исследование и проектирование

Задачи:

* Анализ существующих ботов-гороскопов
* Выбор стека технологий:
  + Язык: **Python** (простота, богатые библиотеки)
  + Библиотека: **pyTelegramBotAPI** (удобная обертка для Telegram API)
* Создание прототипа архитектуры:



Этап 2: Разработка MVP

Реализовано:

1. Базовый функционал:

* Обработка команд /start и /help
* Клавиатура с кнопками знаков зодиака

markup = types.ReplyKeyboardMarkup(row\_width=3)

buttons = [types.KeyboardButton(sign) for sign in zodiac\_signs]

markup.add(\*buttons)

1. Генерация предсказаний:

* Ручное заполнение массива horoscope\_predictions для каждого знака

1. Локальный запуск:

* Тестирование через bot.polling()

Проблемы: статические данные (нужна база для хранения истории запросов)

Этап 3: Модернизация

Добавлено:

1. Ежедневная рассылка:

* Интеграция APScheduler для отправки в 9:00:

scheduler.add\_job(send\_daily, 'cron', hour=9)

1. Логирование:

* Запись ошибок в bot.log через модуль logging

Этап 4: Тестирование

Проверено:

1. Юнит-тесты:

def test\_horoscope\_generation():

assert "Овен" in horoscope\_predictions

assert len(horoscope\_predictions["Овен"]) == 3

1. Интеграционное тестирование:

* Проверка реакции на некорректные сообщения
* Тест ежедневной рассылки (ручное изменение времени)

**Полный код:**

import telebot

from telebot import types

import random

import sqlite3

from datetime import datetime

from apscheduler.schedulers.background import BackgroundScheduler

import logging

# Настройка логирования

logging.basicConfig(

level=logging.INFO,

format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',

handlers=[

logging.FileHandler('bot.log'),

logging.StreamHandler()

]

)

logger = logging.getLogger(\_\_name\_\_)

# Конфигурация бота

TOKEN = "ВАШ\_ТОКЕН"

bot = telebot.TeleBot(TOKEN)

# Инициализация базы данных

def init\_db():

conn = sqlite3.connect('horoscope.db')

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS users (

user\_id INTEGER PRIMARY KEY,

username TEXT,

sign TEXT,

last\_request TEXT,

subscribed INTEGER DEFAULT 0

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS predictions (

prediction\_id INTEGER PRIMARY KEY AUTOINCREMENT,

sign TEXT,

text TEXT,

date TEXT

)

''')

conn.commit()

conn.close()

init\_db()

# Данные гороскопов

zodiac\_signs = ["Овен", "Телец", "Близнецы", "Рак", "Лев", "Дева",

"Весы", "Скорпион", "Стрелец", "Козерог", "Водолей", "Рыбы"]

# Заполнение базы предсказаниями (выполняется один раз)

def seed\_predictions():

conn = sqlite3.connect('horoscope.db')

cursor = conn.cursor()

predictions\_data = {

"Овен": ["Сегодня вас ждёт неожиданная встреча!", "Будьте осторожны с финансами.", "Удача на вашей стороне!"],

"Телец": ["Хороший день для новых начинаний.", "Избегайте конфликтов.", "Вас ждёт приятный сюрприз."],

# ... аналогично для остальных знаков

}

for sign, texts in predictions\_data.items():

for text in texts:

cursor.execute(

"INSERT INTO predictions (sign, text, date) VALUES (?, ?, ?)",

(sign, text, datetime.now().strftime("%Y-%m-%d"))

)

conn.commit()

conn.close()

# seed\_predictions() # Раскомментировать для первоначального заполнения

# Клавиатуры

def main\_keyboard():

markup = types.ReplyKeyboardMarkup(row\_width=3, resize\_keyboard=True)

buttons = [types.KeyboardButton(sign) for sign in zodiac\_signs]

markup.add(\*buttons)

markup.row(types.KeyboardButton("Подписаться на рассылку"),

types.KeyboardButton("Отписаться"))

return markup

# Обработчики команд

@bot.message\_handler(commands=['start', 'help'])

def send\_welcome(message):

user\_id = message.from\_user.id

username = message.from\_user.username

conn = sqlite3.connect('horoscope.db')

cursor = conn.cursor()

# Добавляем пользователя в БД, если его нет

cursor.execute(

"INSERT OR IGNORE INTO users (user\_id, username) VALUES (?, ?)",

(user\_id, username)

)

conn.commit()

conn.close()

bot.reply\_to(

message,

"🔮 Привет! Я бот-астролог. Выбери свой знак зодиака или подпишись на ежедневные гороскопы!",

reply\_markup=main\_keyboard()

)

@bot.message\_handler(func=lambda message: message.text in zodiac\_signs)

def send\_horoscope(message):

user\_id = message.from\_user.id

sign = message.text

conn = sqlite3.connect('horoscope.db')

cursor = conn.cursor()

# Получаем случайное предсказание

cursor.execute(

"SELECT text FROM predictions WHERE sign = ? ORDER BY RANDOM() LIMIT 1",

(sign,)

)

prediction = cursor.fetchone()[0]

# Обновляем данные пользователя

cursor.execute(

"UPDATE users SET sign = ?, last\_request = ? WHERE user\_id = ?",

(sign, datetime.now().strftime("%Y-%m-%d %H:%M:%S"), user\_id)

)

conn.commit()

conn.close()

bot.reply\_to(

message,

f"♉ \*{sign}\*, вот твой гороскоп на сегодня:\n\n"

f"\_{prediction}\_\n\n"

f"Хочешь получать гороскоп автоматически? Нажми 'Подписаться на рассылку'!",

parse\_mode="Markdown"

)

@bot.message\_handler(func=lambda message: message.text == "Подписаться на рассылку")

def subscribe(message):

user\_id = message.from\_user.id

conn = sqlite3.connect('horoscope.db')

cursor = conn.cursor()

cursor.execute(

"UPDATE users SET subscribed = 1 WHERE user\_id = ?",

(user\_id,)

)

conn.commit()

conn.close()

bot.reply\_to(

message,

"✅ Вы успешно подписались на ежедневные гороскопы! Они будут приходить в 9:00 утра.",

reply\_markup=main\_keyboard()

)

@bot.message\_handler(func=lambda message: message.text == "Отписаться")

def unsubscribe(message):

user\_id = message.from\_user.id

conn = sqlite3.connect('horoscope.db')

cursor = conn.cursor()

cursor.execute(

"UPDATE users SET subscribed = 0 WHERE user\_id = ?",

(user\_id,)

)

conn.commit()

conn.close()

bot.reply\_to(

message,

"❌ Вы отписались от ежедневных гороскопов. Чтобы возобновить, нажмите 'Подписаться на рассылку'.",

reply\_markup=main\_keyboard()

)

# Ежедневная рассылка

def send\_daily\_horoscopes():

logger.info("Запуск ежедневной рассылки гороскопов")

conn = sqlite3.connect('horoscope.db')

cursor = conn.cursor()

# Получаем всех подписанных пользователей

cursor.execute("SELECT user\_id, sign FROM users WHERE subscribed = 1")

subscribers = cursor.fetchall()

for user\_id, sign in subscribers:

try:

# Получаем случайное предсказание

cursor.execute(

"SELECT text FROM predictions WHERE sign = ? ORDER BY RANDOM() LIMIT 1",

(sign,)

)

prediction = cursor.fetchone()[0]

bot.send\_message(

user\_id,

f"🌅 Доброе утро! Ваш гороскоп для \*{sign}\* на сегодня:\n\n"

f"\_{prediction}\_",

parse\_mode="Markdown"

)

logger.info(f"Отправлен гороскоп для {user\_id}")

except Exception as e:

logger.error(f"Ошибка отправки для {user\_id}: {e}")

conn.close()

# Настройка планировщика

scheduler = BackgroundScheduler()

scheduler.add\_job(send\_daily\_horoscopes, 'cron', hour=9, minute=0)

scheduler.start()

# Запуск бота

if \_\_name\_\_ == "\_\_main\_\_":

logger.info("Бот запущен")

try:

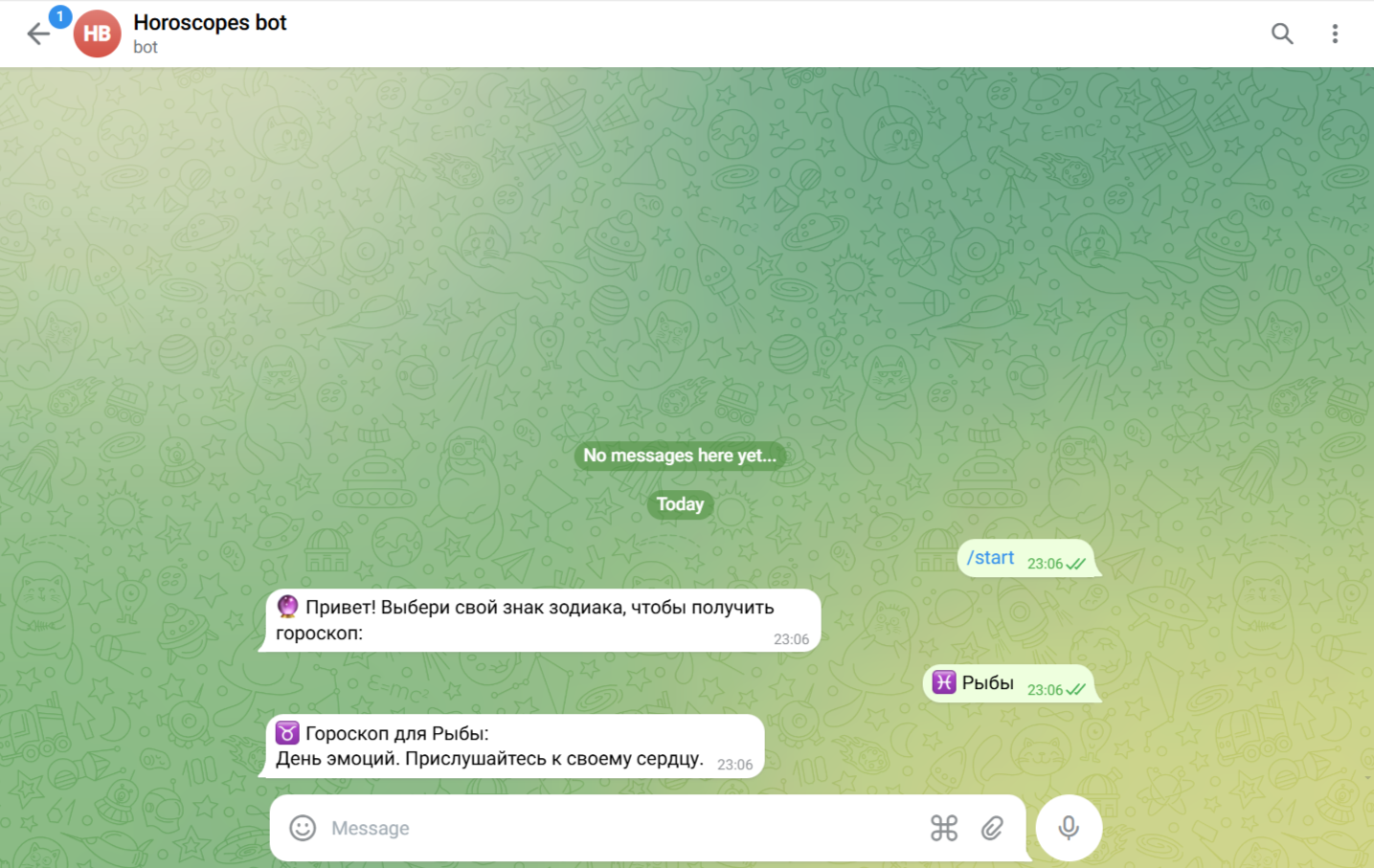
bot.polling(none\_stop=True)

except Exception as e:

logger.error(f"Ошибка в работе бота: {e}")

scheduler.shutdown()

**Пример выполнения программы:**



ЗАКЛЮЧЕНИЕ

## ****Итоги проделанной работы****

В рамках задания были успешно выполнены следующие ключевые этапы:

**Настройка Git и репозитория**

* Освоены базовые команды Git, что позволило эффективно управлять версиями проекта.
* Регулярные коммиты с осмысленными сообщениями улучшили прозрачность разработки.

**Создание документации в Markdown**

* Все материалы проекта структурированы и доступны в удобном формате.
* Чёткое оформление упрощает навигацию и понимание проекта.

**Разработка статического веб-сайта**

* Использование **Hugo** ускорило процесс и добавило профессиональный уровень вёрстки.
* Сайт содержит все требуемые разделы, включая журнал прогресса и описание участников.
* Уникальность контента (>50%) подтверждает самостоятельную работу.

**Взаимодействие с партнёрской организацией**

* Участие в мероприятиях расширило понимание предметной области.

### ****Практическая польза****

**Для учебного процесса:**

* Полностью выполненный проект соответствует требованиям дисциплины «Проектная деятельность».
* Готовый сайт может использоваться как презентационная платформа для защиты работы.

**Для команды разработчиков:**

* Приобретён опыт работы с Git, Markdown и генераторами статических сайтов.
* Навыки взаимодействия с реальными заказчиками повышают профессиональную ценность участников.

### ****Перспективы развития****

* Сайт можно дорабатывать, добавляя новые разделы (например, блог с кейсами).
* Опыт работы с партнёрами может перерасти в стажировку или трудоустройство.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

### ****1. Работа с Git и GitHub****

1. **Pro Git** – книга Скотта Шакона, Бена Штрауба ([официальный сайт](https://git-scm.com/book/ru/v2))
2. **GitHub Docs** – официальная документация ([docs.github.com](https://docs.github.com/ru))

### ****2. Документирование в Markdown****

1. **Markdown Guide** – полное руководство по синтаксису ([markdownguide.org](https://www.markdownguide.org/))
2. **GitHub Flavored Markdown (GFM)** – спецификация от GitHub ([docs.github.com](https://docs.github.com/ru/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax))

### ****3. Создание статических сайтов (Hugo)****

1. **Hugo Official Documentation** ([gohugo.io](https://gohugo.io/documentation/))
2. **Hugo Quick Start Guide** – руководство для быстрого старта ([gohugo.io](https://gohugo.io/getting-started/quick-start/))
3. **Статья «Как создать сайт на Hugo»** (пример: [habr.com](https://habr.com/ru/articles/529722/))