

Департамент профессионального образования Томской области  
ОГБПОУ «ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

МДК.2.2 Инструментальные средства разработки программного обеспечения  
Отчет по выполнению Практической работы №5 «Настройка системы контроля  
версий для управления импортом файлов в репозиторий»

Студент группы 622

\_\_\_\_\_

Силаев В.В.

Преподаватель

\_\_\_\_\_

Демидов Д.Д.

Томск, 2025 г.

Лабораторная работа: Настройка системы контроля версий для управления импортом файлов в репозиторий

### **1. Цель работы:**

Изучить механизмы настройки системы контроля версий для управления включением и исключением файлов и каталогов из репозитория.

Получить практические навыки применения файлов конфигурации игнорирования (например, .gitignore) для управления импортируемыми файлами.

Понять влияние настроек контроля версий на процесс разработки и совместной работы над проектом

### **2. Задачи:**

[Индивидуально:] Определить специфические требования к импорту файлов в репозиторий для конкретной архитектуры проекта и варианта задания (см. ниже).

Изучить возможности системы контроля версий (СКВ) для фильтрации файлов на основе их типов, путей и других критериев (например, Git, Mercurial, Subversion).

Создать и настроить файл конфигурации игнорирования (например, .gitignore для Git) в соответствии с определенными требованиями.

[Индивидуально:] Протестировать настройки, создав тестовые файлы и каталоги, соответствующие архитектуре проекта.

Зафиксировать изменения в репозитории.

[Индивидуально:] Проанализировать результаты тестирования и внести необходимые корректировки в файл конфигурации игнорирования.

**Объем работы:** 2 часа.

**Вариант индивидуального задания:** Магазин бытовой техники

## Основная часть


### Создания репозитория: ( рисунок 1)

#### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \*      Repository name \*



 kivas1k ▾ / 5laba

✔ 5laba is available.

Great repository names are short and memorable. Need inspiration? How about **studious-octo-happiness** ?

Description (optional)

5laba

- ☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 — создание репозитория

**Структура проекта:** Проект построен на базе стандартной структуры Django-приложения без внесения существенных изменений в исходную конфигурацию. Для разработки использовалась интегрированная среда PyCharm, основным язык реализации — Python. Управление версиями кода организовано с помощью Git, что позволяет отслеживать изменения и collaborate эффективно. ( Рисунок 2)

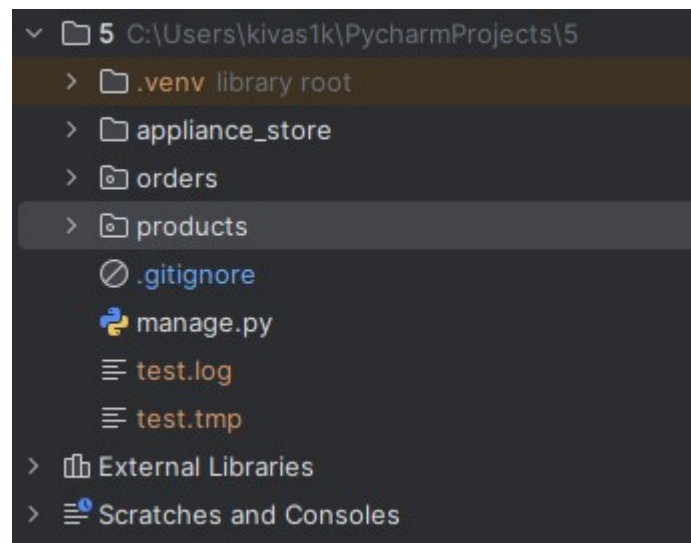
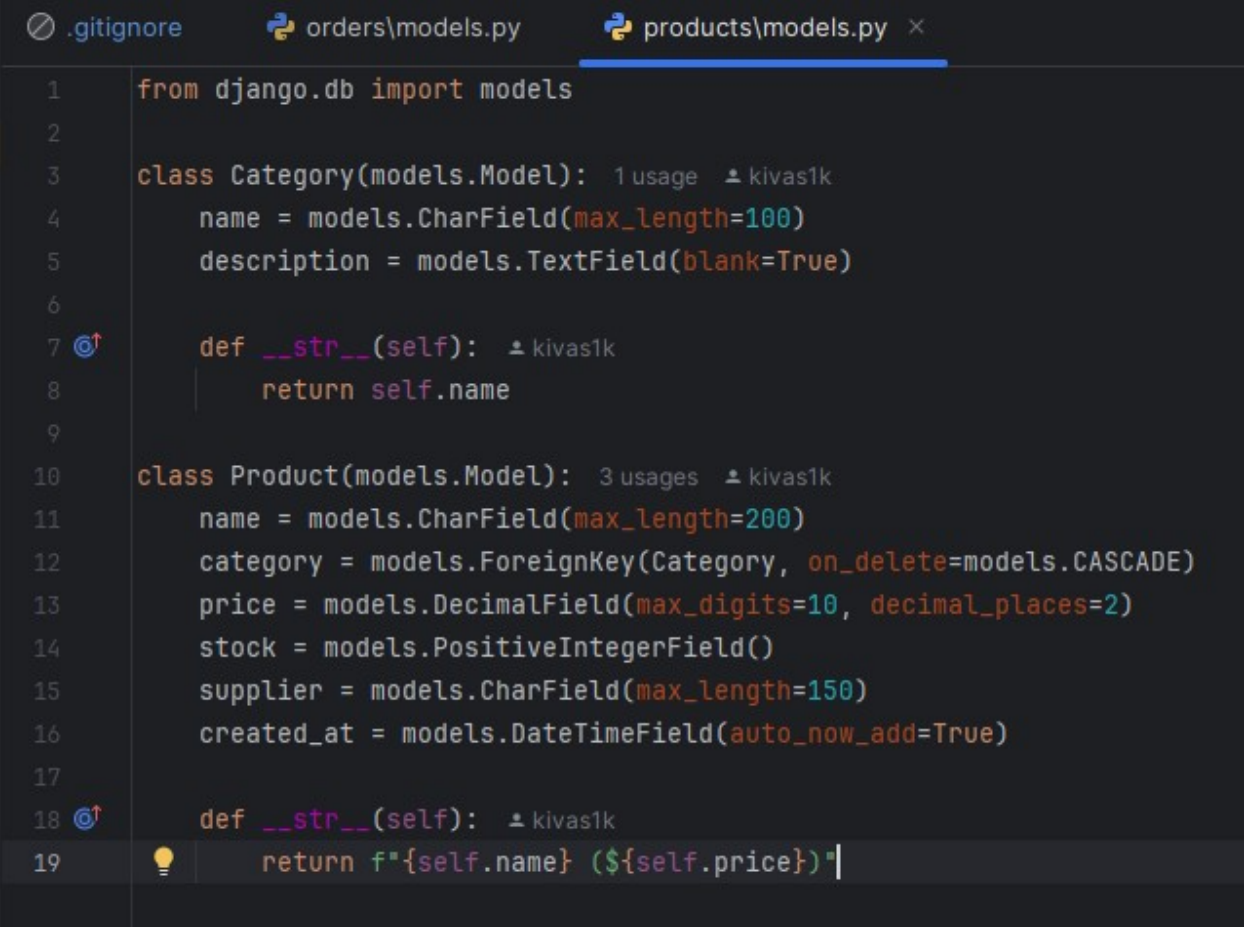


Рисунок 2 — структура проекта

Файлы моделей проекта ( Рисунок 3-4)

A screenshot of a code editor with three tabs: '.gitignore', 'orders\models.py', and 'products\models.py'. The 'products\models.py' tab is active. The code defines two Django models: 'Category' and 'Product'. 'Category' has a 'name' CharField (max\_length=100) and a 'description' TextField (blank=True). 'Product' has a 'name' CharField (max\_length=200), a 'category' ForeignKey to 'Category' (on\_delete=CASCADE), a 'price' DecimalField (max\_digits=10, decimal\_places=2), a 'stock' PositiveIntegerField, a 'supplier' CharField (max\_length=150), and a 'created\_at' DateTimeField (auto\_now\_add=True). Both models have a '\_\_str\_\_' method that returns the object's name. The code is written in Python with syntax highlighting. Line numbers 1 through 19 are visible on the left side of the editor.

```
1 from django.db import models
2
3 class Category(models.Model): 1 usage 1 kivas1k
4     name = models.CharField(max_length=100)
5     description = models.TextField(blank=True)
6
7     def __str__(self): 1 kivas1k
8         return self.name
9
10 class Product(models.Model): 3 usages 1 kivas1k
11     name = models.CharField(max_length=200)
12     category = models.ForeignKey(Category, on_delete=models.CASCADE)
13     price = models.DecimalField(max_digits=10, decimal_places=2)
14     stock = models.PositiveIntegerField()
15     supplier = models.CharField(max_length=150)
16     created_at = models.DateTimeField(auto_now_add=True)
17
18     def __str__(self): 1 kivas1k
19     return f"{self.name} (${self.price})"
```

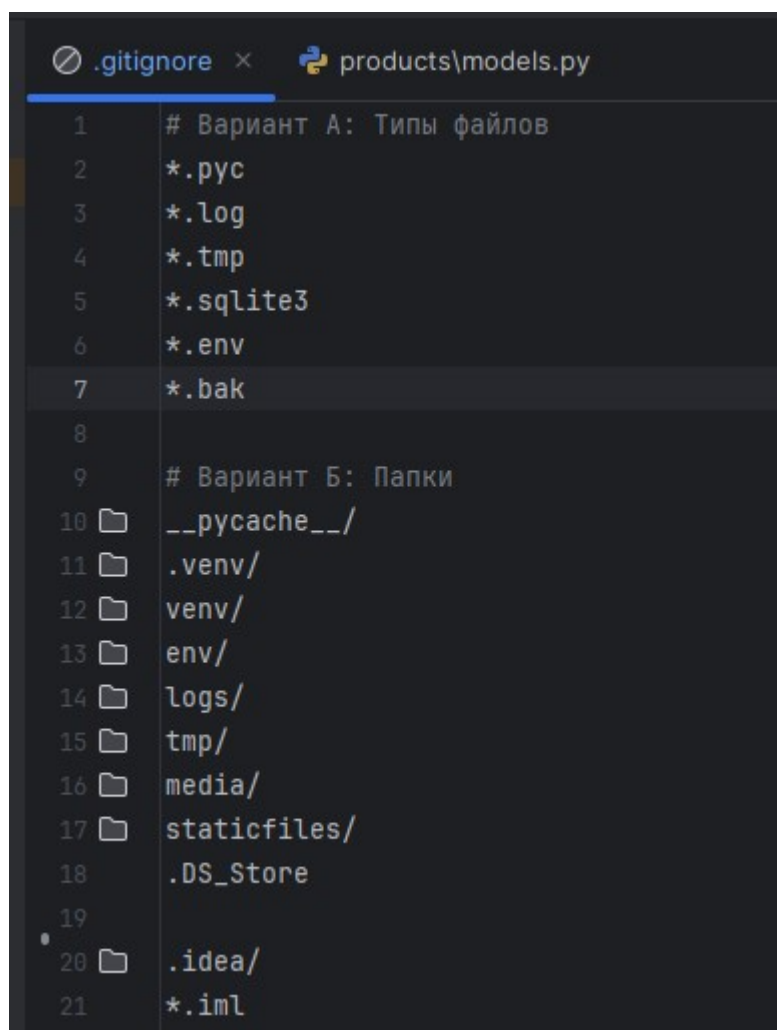
Рисунок 3 — модель для продуктов в проекте

```
.gitignore  orders\models.py  products\models.py

1  from django.db import models
2  from products.models import Product
3
4  class Customer(models.Model): 1 usage  👤 kivas1k
5      name = models.CharField(max_length=100)
6      phone = models.CharField(max_length=20)
7      email = models.EmailField(blank=True)
8      address = models.TextField()
9
10  def __str__(self):  👤 kivas1k
11      return self.name
12
13  class Order(models.Model): 1 usage  👤 kivas1k
14      customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
15      products = models.ManyToManyField(Product, through='OrderItem')
16      total_price = models.DecimalField(max_digits=10, decimal_places=2)
17      created_at = models.DateTimeField(auto_now_add=True)
18
19  class OrderItem(models.Model):  👤 kivas1k
20      order = models.ForeignKey(Order, on_delete=models.CASCADE)
21      product = models.ForeignKey(Product, on_delete=models.CASCADE)
22  quantity = models.PositiveIntegerField()
```

Рисунок 4 — модель для покупок в проекте

В качестве вариантов задания были выбраны А и Б. ( Рисунок 5 )

A screenshot of a code editor with a dark theme. The editor has two tabs at the top: ".gitignore" (selected) and "products\models.py". The ".gitignore" file contains two sections of ignore rules. The first section, starting at line 1, is titled "# Вариант А: Типы файлов" and lists file extensions: \*.pyc, \*.log, \*.tmp, \*.sqlite3, \*.env, and \*.bak. The second section, starting at line 9, is titled "# Вариант Б: Папки" and lists directories: \_\_pycache\_\_/, .venv/, venv/, env/, logs/, tmp/, media/, staticfiles/, .DS\_Store, .idea/, and \*.iml. Line numbers 1 through 21 are visible on the left side of the editor.

```
1 # Вариант А: Типы файлов
2 *.pyc
3 *.log
4 *.tmp
5 *.sqlite3
6 *.env
7 *.bak
8
9 # Вариант Б: Папки
10 __pycache__/
11 .venv/
12 venv/
13 env/
14 logs/
15 tmp/
16 media/
17 staticfiles/
18 .DS_Store
19
20 .idea/
21 *.iml
```

Рисунок 5 — файл .gitignore

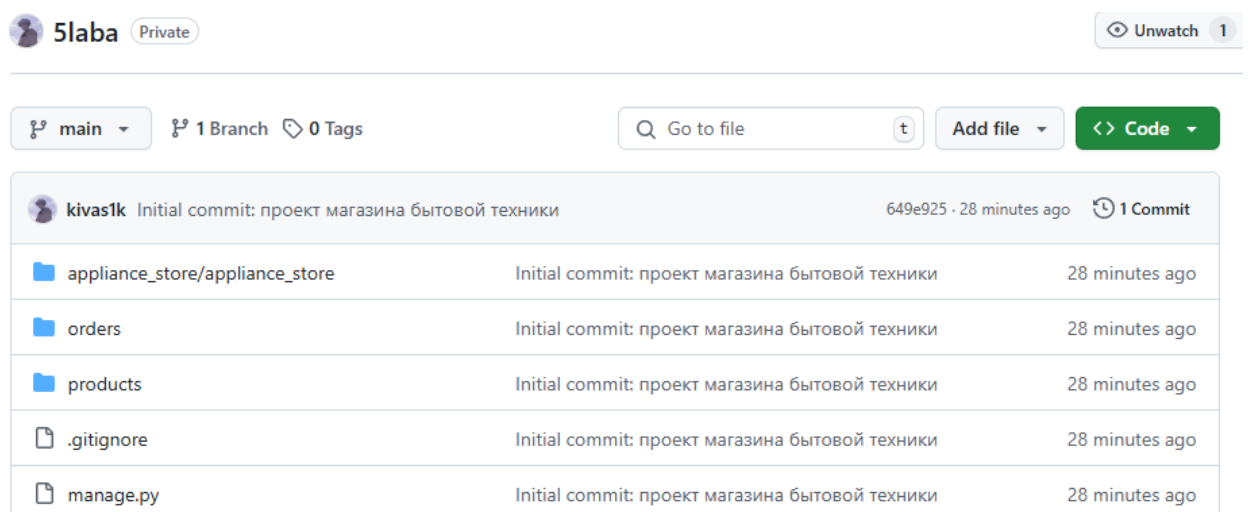
Первый коммит и тестирование игнорирования выбранных файлов по заданию. ( рисунок 6)

```
(.venv) PS C:\Users\kivas1k\PycharmProjects\5> git remote add origin https://github.com/kivas1k/5laba.git
error: remote origin already exists.
(.venv) PS C:\Users\kivas1k\PycharmProjects\5> git branch -M main
(.venv) PS C:\Users\kivas1k\PycharmProjects\5> git push -u origin main
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 12 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (19/19), 4.09 KiB | 2.05 MiB/s, done.
Total 19 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/kivas1k/5laba.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
(.venv) PS C:\Users\kivas1k\PycharmProjects\5> git status --ignored
On branch main
Your branch is up to date with 'origin/main'.

Ignored files:
(use "git add -f <file>..." to include in what will be committed)
.idea/
.venv/
test.log
test.tmp
```

Рисунок 6 — первый коммит и ввод команды для поиска игнорируемых файлов

Ссылка на репозиторий: <https://github.com/kivas1k/5laba> (рисунок 7)



5laba Private Unwatch 1

main 1 Branch 0 Tags Go to file Add file Code

File	Commit Message	Time
appliance_store/appliance_store	Initial commit: проект магазина бытовой техники	28 minutes ago
orders	Initial commit: проект магазина бытовой техники	28 minutes ago
products	Initial commit: проект магазина бытовой техники	28 minutes ago
.gitignore	Initial commit: проект магазина бытовой техники	28 minutes ago
manage.py	Initial commit: проект магазина бытовой техники	28 minutes ago

Рисунок 7 — страница репозитория



### **Итоговый вывод:**

В процессе работы удалось освоить базовые принципы настройки системы контроля версий для управления файлами в репозитории, что позволило исключить попадание в него нежелательных данных, таких как автоматически генерируемые фреймворком артефакты (кеш, временные файлы), медиаматериалы, используемые в разработке, а также служебные файлы среды разработки и виртуального окружения. Текущие настройки эффективно решают задачу фильтрации, сохраняя репозиторий чистым и сфокусированным на исходном коде. Регулярная актуализация `.gitignore` станет ключевыми шагами для поддержания оптимальной структуры репозитория и предотвращения его «засорения» техническими данными.