Факультет компьютерных наук

Департамент программной инженерии

Москва 2026

Веб-сервис автопроверки работ в формате ipynb

Notebook Grader: проверка задач и отчёт для преподавателя

Индивидуальный программный проект

Исполнитель: Бюрчиев Тимур Зольванович, БПИ248
Научный руководитель: Жукова Галина Николаевна (Доцент ДПИ ФКН, Кандидат наук)

Проблема: ручная проверка .ipynb-работ

Дисциплины по Python, анализу данных, машинному обучению используют Jupyter Notebook как основной формат заданий. На потоках в десятки и сотни студентов проверка превращается в массовую механическую работу: открыть ноутбук, запустить ячейки, сверить с эталоном, поставить балл, перенести в ведомость.

Кол-во работ	Ручная проверка	Автоматическая	Экономия
50	2 ч 30 мин	≈ 4 мин	2 ч 26 мин
100	5 ч	≈ 8 мин	4 ч 52 мин
150	7 ч 30 мин	≈ 12 мин	7 ч 17 мин

Эффект пропорционален размеру потока: чем больше работ, тем больше сэкономленное время преподавателя.

Цель и задачи проекта

Цель: разработать веб-сервис для автопроверки .ipynb-работ и формирования отчёта для преподавателя.

Задачи:

1. Проанализировать существующие решения проверки работ.
2. Спроектировать архитектуру веб-сервиса.
3. Реализовать разбор .ipynb и сопоставление задач по тегам.
4. Реализовать три способа автоматической проверки.
5. Обеспечить изолированный запуск кода в Docker.
6. Реализовать интерфейс преподавателя и экспорт результатов.

```
k, m = symbols('k m')
expr3 = (cos(pi * k ) ** 2 + (3 - sin(pi * k)) ** 2 - 10) ** 2\
+ 36 * cos(pi * k) ** 2
display(expr3, simplify(expr3))
```

$$\left((3 - \sin(\pi k))^2 + \cos^2(\pi k) - 10\right)^2 + 36\cos^2(\pi k)$$

$$36$$

Задание 4.

Упростите выражение $\frac{(a+b+c+d)^2-(a+c)^2-(b+d)^2}{a+c}$

```
a, b, c, d = symbols('a b c d')
expr4 = ((a + b + c + d) ** 2 - (a + c) ** 2 - (b + d) ** 2) / (a + c)
simplify(expr4).factor()
```

$$2(b + d)$$

Пример ноутбука

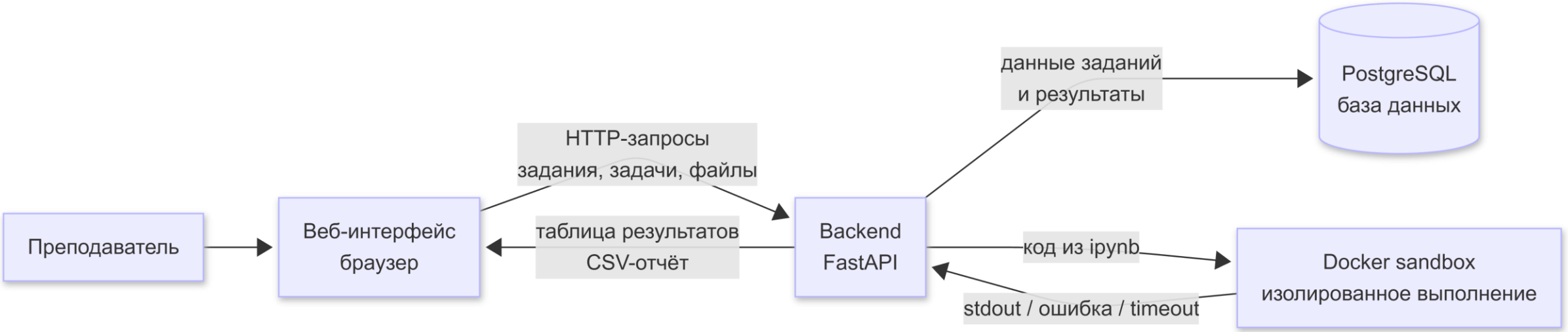
Анализ существующих решений

Решение	Сильная сторона	Ограничение для целевого сценария
Контест-системы (Я.Контест, Codeforces)	Подходят для задач с вводом/выводом и соревновательного формата	Требуют переноса задания в формат платформы; не работают с готовыми ноутбук-файлами
nbgrader	Специализирован для Jupyter Notebook	Требует развёртывания и поддержки инфраструктуры JupyterHub
Локальные скрипты	Гибкость, быстрое решение под конкретное задание	Нет веб-интерфейса, хранения результатов, единого сценария работы
Notebook Grader	Объединяет загрузку .ipynb, настройку проверки, изолированный запуск и CSV-отчёт в одном сервисе	Поддерживается только Python; ориентирован на учебный сценарий массовой проверки

Пользовательский сценарий: работа преподавателя с сервисом

1. Настроить задание и задачи вручную или по шаблону
2. Загрузить до 10 .ipynb-файлов
3. Запустить проверку: разбор ipynb и выполнение кода
4. Посмотреть результаты: баллы, оценка, статус
5. Скачать CSV ведомость по группе

Архитектура веб-сервиса



Формат входных данных: размеченный notebook

Сервис распознаёт три служебных элемента в ipynb-файле

Маркер студента
markdown-ячейка

```
<!-- STUDENT_INFO -->
ФИО: Иванов И.И.
Группа: БПИ241
```

→ ФИО и группа для отчёта

Setup-ячейка
code, тег `setup`

```
import numpy as np
import pandas as pd
```

→ общий код перед каждой задачей

Task-ячейка
code, тег `task:<код>`

```
def solve(x):
    return x ** 2
print(solve(5))
```

→ сопоставление с задачей задания по коду

Три способа проверки задач

Преподаватель выбирает тип проверки для каждой задачи

Тип	Что задаёт преподаватель	Как проверяется	Когда применяется
answer	Эталонный ответ	Запуск кода, сравнение stdout с ожидаемым выводом	Задачи с одним известным выводом
tests	Набор тест-кейсов	Для каждой пары stdin - stdout выполняется отдельная проверка	Задачи, где решение проверяется на нескольких входах
reference_assert	Проверочный Python-код	Код преподавателя дописывается к решению; исключение – 0 б.	Проверка функций, переменных, структур данных

Пример reference_assert:

```
assert callable(solve)
assert solve(5) == 25
assert solve(-3) == 9
```

Изолированное выполнение студенческого кода

Код из notebook запускается в отдельном Docker-контейнере, а не в основном backend-процессе.

Ограничение	Значение	Что предотвращает
Сеть	отключена	внешние запросы
Файловая система	read-only, /tmp 64 МБ	изменение окружения, запись файлов
Память	1 ГБ	чрезмерное потребление памяти
CPU	1 ядро	перегрузку сервера
Время	30 секунд	бесконечные циклы

sandbox только выполняет код; решение о баллах принимает модуль проверки.

Технологический стек проекта

Пользовательский интерфейс
HTML · Bootstrap · JavaScript

Формы, таблицы, модальные окна и загрузка файлов без тяжёлого frontend-фреймворка

Backend
Python · FastAPI · Pydantic

REST API, обработка пользовательских действий и валидация входных данных

Хранение данных
PostgreSQL · SQLAlchemy · Alembic

Связанные сущности: преподаватель, задание, задача, работа студента, результат проверки

Работа с notebook-файлами
Nbformat

Чтение структуры .ipynb, markdown/code-ячеек и служебных тегов

Изолированная проверка
Docker · Docker SDK for Python

Запуск студенческого кода в отдельном контейнере с ограничениями

Демонстрация

Результаты

1. Проведён анализ существующих решений.
2. Спроектирована архитектура веб-сервиса.
3. Реализован разбор .ipynb-файлов по служебным тегам.
4. Добавлены три способа проверки: answer, tests, reference_assert.
5. Настроен изолированный запуск студенческого кода в Docker-контейнере с ограничениями ресурсов.
6. Реализован интерфейс преподавателя: управление заданиями, загрузка работ, просмотр результатов и экспорт CSV-отчёта.

GitHub репозиторий

Веб-сайт

Планы развития

1. Производительность
    - Параллельная обработка работ через очередь задач
    - Снятие ограничения на 10 файлов за одну загрузку

2. Качество проверки
    - Расширение sandbox-окружения под задачи анализа данных, математики и машинного обучения
    - Развитие проверок для функций, структур данных и результатов вычислений

3. Пользовательский сценарий
    - Личный кабинет студента
    - Самостоятельная сдача работ и просмотр детализации по задачам

Список использованных источников

1. Moodle [Электронный ресурс]. URL: https://moodle.org/ (дата обращения: 20.05.2026)
2. Canvas LMS [Электронный ресурс]. URL: https://www.instructure.com/canvas (дата обращения: 20.05.2026)
3. Яндекс Контест [Электронный ресурс]. URL: https://contest.yandex.ru/ (дата обращения: 20.05.2026)
4. nbgrader documentation [Электронный ресурс]. URL: https://nbgrader.readthedocs.io/ (дата обращения: 20.05.2026)
5. FastAPI documentation [Электронный ресурс]. URL: https://fastapi.tiangolo.com/ (дата обращения: 20.05.2026)
6. PostgreSQL documentation [Электронный ресурс]. URL: https://www.postgresql.org/docs/ (дата обращения: 20.05.2026)
7. nbformat documentation [Электронный ресурс]. URL: https://nbformat.readthedocs.io/ (дата обращения: 20.05.2026)
8. Docker SDK for Python documentation [Электронный ресурс]. URL: https://docker-py.readthedocs.io/ (дата обращения: 20.05.2026)

Спасибо за внимание!

tzbiurchiev@edu.hse.ru