УДК 004.584

АВТОМАЗИЦИЯ ПРОЦЕССА РАБОТЫ СО ШКОЛЬНИКАМИ ПО ПРОФЕССИОНАЛЬНОМУ ОРИЕНТИРОВАНИЮ НА ОСНОВЕ ВЕБ ТЕХНОЛОГИИ

Кобелев М.В., студент гр. ИТб-221, IV курс Асанов С.А., старший преподаватель кафедры ИиАПС
Кузбасский государственный технический университет имени Т. Ф. Горбачева
г. Кемерово

Профессиональная ориентация школьников и абитуриентов является важной задачей для системы образования и рынка труда. Осознанный выбор профессии на ранних этапах позволяет снизить риск разочарования в будущей специальности, повысить мотивацию к обучению и обеспечить экономику квалифицированными кадрами. В настоящее время существует множество разрозненных источников информации о профессиях, вузах и требованиях к поступлению, что затрудняет для школьников формирование целостной картины.

Для решения данной проблемы разрабатывается веб-приложение, направленное на автоматизацию процесса профориентационной работы. Приложение позволяет школьникам и абитуриентам проходить диагностические тесты, вести индивидуальную карточку интересов и достижений, получать персонализированные рекомендации по выбору профессии, а также знакомиться с актуальной информацией об учебных заведениях и предприятиях региона.

В данной статье рассматривается архитектура и функциональные возможности разрабатываемого веб-приложения, а также подходы к его реализации с использованием технологий Python, Django и PostgreSQL.

В настоящее время профориентационная работа в школах часто ведется в формате разовых мероприятий: классных часов, встреч с представителями вузов или тестирования на бумажных носителях. Такой подход не позволяет накапливать данные о предпочтениях учеников в динамике, не обеспечивает персонализированных рекомендаций и не дает возможности оперативно знакомить учащихся с актуальными изменениями на рынке труда.

Разрабатываемое приложение призвано автоматизировать ключевые этапы профориентационной работы и объединить в единой цифровой среде всех участников процесса: школьников, педагогов, представителей учебных заведений и работодателей. Система получила рабочее название «ПрофНавигатор» и включает следующие механизмы:

- Прохождение психологических и профессиональных опросов и тестов;
- Ведение личной карточки учащегося с интересами, увлечениями и

достижениями;

- Формирование списка подходящих профессий на основе результатов тестов и данных карточки;
- Просмотр детальной информации о профессиях (необходимые школьные предметы, описание, востребованность);
- Фильтрация профессий по региону востребованности;
- Предоставление списка вузов и колледжей, где ведется обучение по выбранной профессии;
- Размещение предприятиями полезной информации в формате коротких видео, фото и аудиоматериалов;
- Запись школьников на экскурсии на предприятия.

В данной статье рассмотрены архитектурные решения, позволяющие реализовать перечисленный функционал в виде веб-приложения.

Веб-приложение «ПрофНавигатор» не требует установки на рабочее место пользователя и доступно через браузер при наличии интернет-соединения. Оно разработано на языке Python с использованием фреймворка Django. В качестве системы управления базами данных используется PostgreSQL, что обеспечивает надежное хранение данных о пользователях, результатах тестов, профессиях, учебных заведениях и предприятиях.

Программа работает на сервере непрерывно, обрабатывая HTTP-запросы от пользователей. Взаимодействие с базой данных осуществляется по запросу представлений (views), которые реализуют бизнес-логику: обработку результатов тестов, фильтрацию профессий, формирование рекомендаций.

На рисунке 1 представлена диаграмма деятельности процесса формирования рекомендаций по выбору профессии. Пользователь последовательно проходит тестирование, заполняет личную карточку, после чего система агрегирует полученные данные, сопоставляет их с базой знаний о профессиях и выдает персонализированный список рекомендаций.

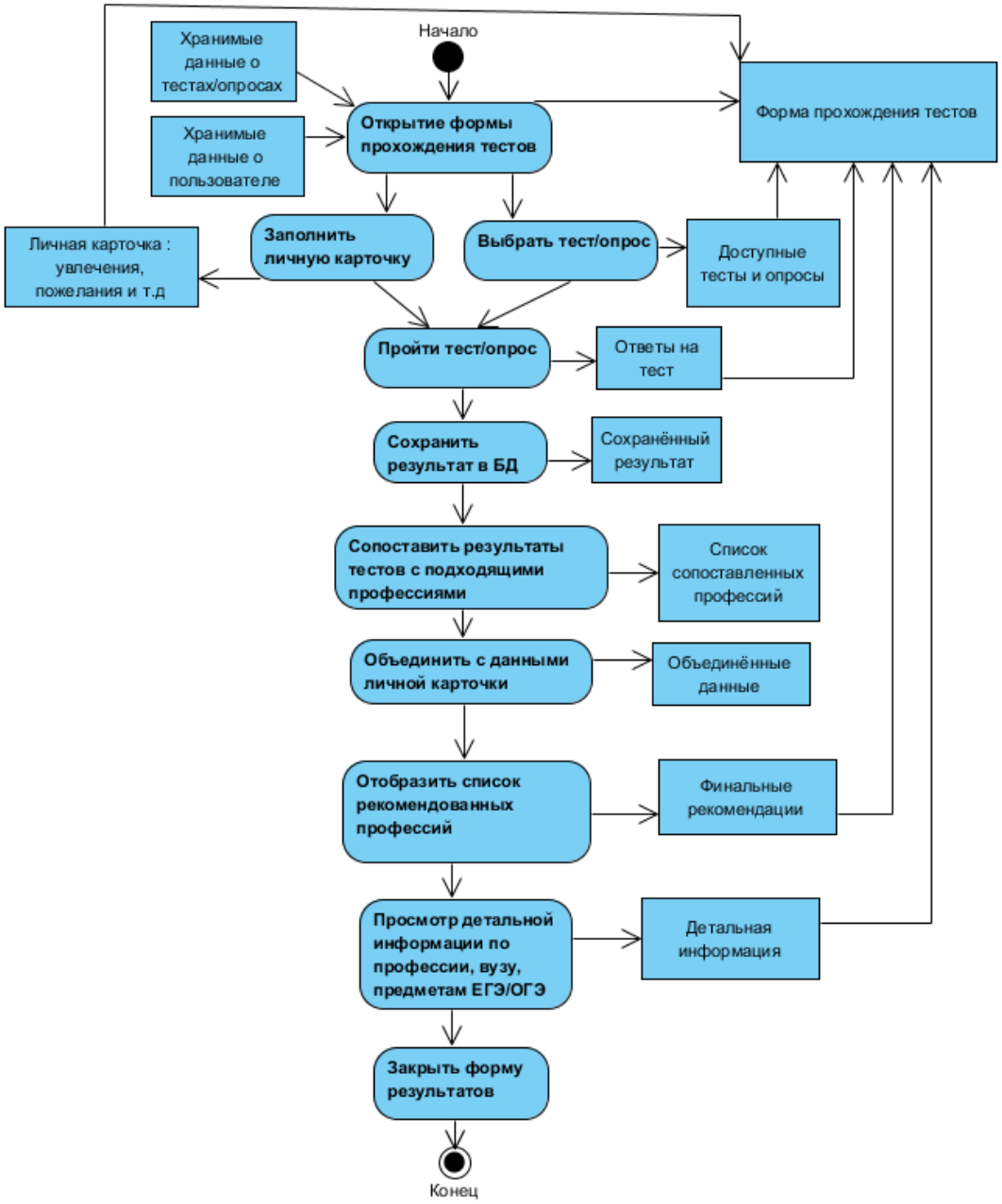

Рисунок 1 – Диаграмма деятельности процесса формирования рекомендаций по выбору профессий

Для реализации механизма размещения информационного контента компаниями разработан модуль управления мультимедийными материалами. На рисунке 2 представлена диаграмма деятельности процесса размещение контента компанией.

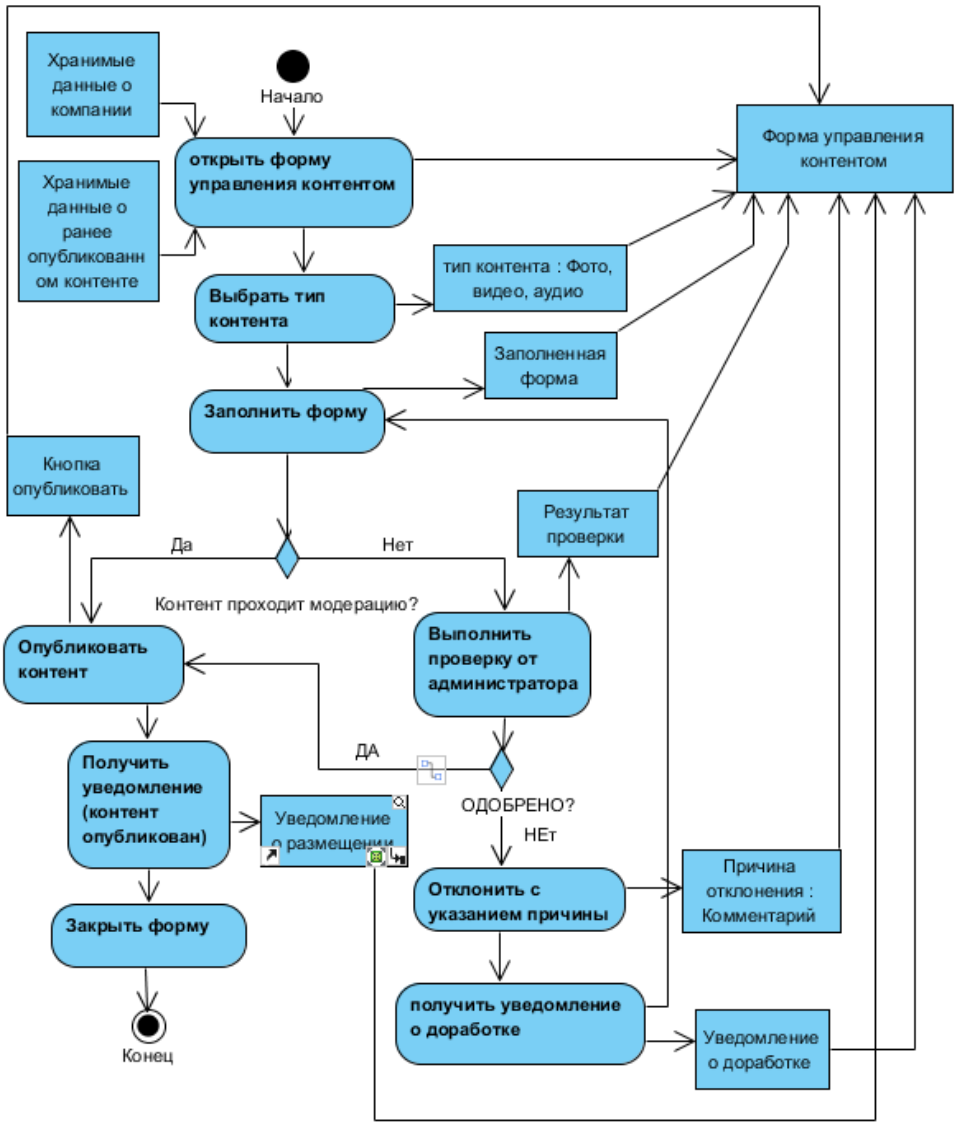

Рисунок 2 – Диаграмма деятельности процесса размещение контента компанией

Для создания web-приложения используем технологию Django и язык Python.

Данный фреймворк предоставляет встроенную ORM (Object-Relational Mapping), что позволяет описывать структур базы данных в виде Python-классов и минимизировать написание «сырых» SQL-запросов. Взаимодействие с PostgreSQL осуществляется через миграции, которые автоматически синхронизируют модели данных с таблицами в СУБД.

Архитектура приложения строится на классическом паттерне MVC, который в Django реализован как MVT (Model-View-Template). Модели описывают сущности предметной области: «Пользователь», «Профессия», «Вопрос», «РезультатТеста», «КарточкаУчащегося», «УчебноеЗаведение», «Предприятие», «Экскурсия». Представления (views) обрабатывают HTTP-запросы, выполняют бизнес-логику (расчет соответствия профессии, фильтрация, сохранение данных) и возвращают ответ. Шаблоны (templates) генерируют HTML-страницы для взаимодействия с пользователем через браузер.

Особое внимание при проектировании уделено разграничению прав

доступа. В системе предусмотрены следующие роли:

Школьник / Студент – проходит тесты, заполняет карточку, просматривает рекомендации, ищет профессии и учебные заведения, записывается на экскурсии.

Педагог – просматривает результаты профориентационной работы своих учеников, получает агрегированную статистику.

Представитель предприятия – размещает информацию о предприятии (видео, фото, аудио), публикует доступные экскурсии, взаимодействует с записями.

Администратор – управляет списком профессий, учебных заведений, вопросов тестов, назначает роли пользователям, анализирует общую статистику.

Для реализации оповещений в реальном времени (например, подтверждение записи на экскурсию, уведомление педагога о прохождении учеником теста) в веб-приложении используются механизмы Django Channels и WebSockets. Это позволяет отправлять уведомления без необходимости перезагрузки страницы.

Модель данных для учета результатов тестирования и хранения карточки учащегося продемонстрирована на рисунке 3. Данная модель используется представлением recommendation_view для формирования списка рекомендуемых профессий. При загрузке страницы выполняется запрос к БД, агрегирующий данные из таблиц результатов тестов и личной карточки. Алгоритм сопоставления взвешивает ответы пользователя по ключевым критериям (интересы, школьные успехи, предпочитаемые предметы) и находит наиболее релевантные профессии.

```python
class Card(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='card')
    interests = models.TextField(verbose_name='Увлечения и интересы', blank=True)
    achievements = models.TextField(verbose_name='Достижения', blank=True)
    preferred_subjects = models.CharField(max_length=200, verbose_name='Предпочитаемые школьные предметы', blank=True)
    future_plans = models.TextField(verbose_name='Пожелания по будущей профессии', blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"Карточка {self.user.username}"


class TestResult(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name='test_results')
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    selected_option = models.ForeignKey(AnswerOption, on_delete=models.CASCADE, null=True)
    answer_text = models.TextField(blank=True, verbose_name='Текстовый ответ')
    taken_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        unique_together = ['user', 'question']

    def __str__(self):
        return f"{self.user.username} - {self.question.text[:30]}"
```

Рисунок 3 – Модель данных для учета результатов тестирования и карточки учащегося.

Функционал записи на экскурсии реализован через отдельное представление excursion_booking_view. Школьник просматривает список доступных экскурсий, выбирает предприятие и удобное время. После этого

представитель предприятия получает уведомление на своей рабочей панели. Для предотвращения конфликтов при записи в Django-приложении реализован механизм блокировок на уровне транзакций базы данных.

Тестирование веб-приложения проводилось в среде разработки с использованием встроенного сервера Django, а затем с использованием Gunicorn в качестве WSGI-сервера и Nginx для проксирования запросов. База данных PostgreSQL разворачивалась на отдельном сервере. Результаты тестирования показали, что при одновременной работе до 100 пользователей (школьников, педагогов, представителей предприятий) среднее время отклика интерфейса не превышает 1,5 секунд, а формирование персонализированного списка рекомендаций занимает менее 2 секунд. Уведомления через WebSockets доходят за 0,3–0,7 секунды.

Благодаря внедрению веб-приложения «ПрофОриентир» в школах и учреждениях среднего профессионального образования достигается непрерывность профориентационной работы, обеспечивается персонализированный подход к каждому учащемуся, сокращаются временные и организационные затраты педагогов. Использование Django и PostgreSQL обеспечивает масштабируемость: при увеличении числа пользователей или расширении базы профессий достаточно добавить вычислительные ресурсы серверу без изменения логики приложения.

Список литературы:

1. Документация Django [Электронный ресурс]. – URL: https://docs.djangoproject.com/en/6.0/ (дата обращения: 29.03.2026)
2. Документация PosgtreSQL [Электронный ресурс] – URL: https://www.postgresql.org/docs/ (дата обращения: 29.03.2026)
3. Моделирование на языке UML в среде Visual Paradigm [Электронный ресурс] – URL: http://sp.cs.msu.ru/ooap/exer2017.html (дата обращения: 29.03.2026)