Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Работа выполнена в СПБ «DeCode»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС

_____ Е.М. Димитриади
(подпись)

«____ » _____ 20___ г.

Декан ФКТ

_____И.А. Трещев
(подпись)

«____ » _____ 20___ г.

УТВЕРЖДАЮ

И. о. проректора
по научной работе

_____А.В. Космынин
(подпись)

«____ » _____ 20___ г.

Разработка веб-сервиса создания электронных тренингов
Комплект *конструкторской / проектной* документации

Руководитель СПБ _____ *Е.Э. Шаповалов*
(подпись, дата)

Руководитель проекта _____ *Е.Б. Абарникова*
(подпись, дата)

Комсомольск-на-Амуре 2026

Карточка проекта

Название	*Проектирование веб-сервиса создания электронных тренингов*
Тип проекта	Тип проекта: техническое творчество (инициативный)
Исполнители	Студенты Т.С. Сидорин – 2ИТб-1, К.М. Черненко – 2ИТб-1
Срок реализации	30.11.2025 по 31.05.2026

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ЗАДАНИЕ
на разработку

Название проекта: Проектирование веб-сервиса создания электронных тренингов

Назначение: Назначение проекта по созданию веб-сервиса "Конструктор тренингов" заключается в предоставлении пользователям удобной и функциональной платформы для самостоятельного создания и проведения тренингов, семинаров, мастер-классов и других образовательных мероприятий.

Область использования: учебные заведения, преподаватели и студенты могут использовать сервис для создания и проведения образовательных мероприятий, таких как семинары, мастер-классы, конференции

Функциональное описание проекта:

- Регистрация и авторизация пользователей: предоставление возможности зарегистрироваться и войти в свой личный кабинет на сервисе.
- Создание и редактирование тренингов -
- Интерактивное создание шагов тренинга

Техническое описание устройства:

Требования:

- авторизация;
- регистрация;
- Создание и редактирование тренинга;
-Удаление тренинга и обновление информации о тренинге;

Нефункциональные требования:

- Надежность: сервис должен быть надежным и обеспечивать бесперебойную работу, минимизируя риски сбоев и потерь данных.

Безопасность: сервис должен обеспечивать безопасность данных пользователей, включая персональную информацию, контент тренингов и отзывы. Для этого необходимо использовать современные методы защиты данных, такие как шифрование, авторизация и аутентификация.

Масштабируемость: сервис должен быть масштабируемым и способным поддерживать растущее количество пользователей и тренингов, а также обеспечивать высокую производительность при высокой нагрузке;

План работ:

Наименование работ	Срок
Исследование требований	30.11.2025-30.12.2025
Создание прототипа	11.01.2026-25.01.2026
Разработка клиентской части	05.02.2026-10.03.2026
Разработка серверной части	11.03.2026-20.04.2026
Тестирование и отладка	26.04.2026-25.05.2026
Публикация	31.05.2026

Комментарии:

Разработка проекта велась с использованием agile-методологий, у каждого члена команды была своя задача, благодаря этому выполнение некоторых этапов велось параллельно

Перечень графического материала:

1. Принципиальная схема;
2. Чертежи изделия (или трехмерные модели изделия);
3. Внешний вид изделия;
4. Блок-схема алгоритмов (при наличии управляющих программ);

Руководитель проекта _____ *Е.Б. Абарникова*

(подпись, дата)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ПАСПОРТ

Разработка веб-сервиса создания электронных тренингов

Руководитель проекта _____ *Е.Б. Абарникова*
(подпись, дата)

Комсомольск-на-Амуре 2026

Содержание

1 Общие положения

Настоящий паспорт является документом, предназначенным для ознакомления с основными техническими характеристиками, устройством, правилами установки и эксплуатации проекта «Интерактивный веб-сервис Конструктор тренингов» (далее «сервис»).

1.1 Наименование изделия

Полное наименование изделия – «Интерактивный веб-сервис Конструктор тренингов».

1.2 Наименования документов, на основании которых ведется проектирование изделия

Разработка веб-сервиса создания электронных тренингов осуществляется на основании требований и положений следующих документов:

- задание на разработку.

1.3 Перечень организаций, участвующих в разработке изделия

Заказчиком проекта «Интерактивный веб-сервис Конструктор тренингов» является Федеральное государственное бюджетное образовательное учреждение высшего образования «Комсомольский-на-Амуре государственный университет» (далее заказчик), находящийся по адресу: 681013, Хабаровский край, г. Комсомольск-на-Амуре, Ленина пр-кт., д. 17.

Исполнителем проекта «Интерактивный веб-сервис Конструктор тренингов» являются Конструкторы студенческого конструкторского студенческого проектного бюро «DeCode» (далее СПБ «DeCode»), студенты группы 2ИТб-1 Т.С. Сидорин, К.М. Черненко.

1.4 Сведения об использованных при проектировании нормативно-технических документах

При проектировании использованы следующие нормативно-технические документы:

ГОСТ 2.001-2013. Единая система конструкторской документации. Общие положения.

ГОСТ 2.102-2013. Единая система конструкторской документации. Виды и комплектность конструкторских документов.

ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам.

ГОСТ 2.610-2006. Единая система конструкторской документации. Правила выполнения эксплуатационных документов.

ГОСТ 2.004-88. Единая система конструкторской документации. Общие требования к выполнению конструкторских технологических документов на печатающих и графических устройствах вывода ЭВМ.

ГОСТ 2.051-2006. Единая система конструкторской документации. Электронные документы. Общие положения.

ГОСТ 2.052-2006. Единая система конструкторской документации. Электронная модель изделия. Общие положения.

ГОСТ 2.601-2013. Единая система конструкторской документации. Эксплуатационные документы.

2 Техническое задание

2.1 Наименование проекта

Разработка интерактивного сервиса «Конструктор тренингов».

2.2 Цель проекта

Целью проекта по созданию веб-сервиса «Конструктор тренингов»является предоставление пользователям удобной и функциональной платформы для самостоятельного создания, продвижения и проведения тренингов, семинаров, мастер-классов и других образовательных мероприятий.

2.3 Требования к функциональности

Требования к сервису в целом. Сервис представляет собой веб-платформу для организации и проведения тренингов. Основной задачей платформы является централизация процессов создания, поиска, регистрации и участия в обучающих мероприятиях для корпоративных или образовательных целей. Платформа должна упростить взаимодействие между организаторами (создателями) тренингов и их участниками, а также автоматизировать рутинные задачи по управлению мероприятиями.

Система должна выполнять следующие задачи:

- Управление учетными записями пользователей. Система должна обеспечивать возможность регистрации, аутентификации и управления профилем для всех пользователей.
- Создание и управление тренингами. Пользователи» должны иметь возможность создавать новые тренинги, указывая их название, описание, дату, время, также редактировать и удалять свои тренинги.
- Поиск и фильтрация тренингов. Все пользователи должны иметь возможность просматривать каталог доступных тренингов с функциями

поиска по ключевым словам и фильтрации по категориям, датам или создателям.

- Личный кабинет пользователя. Каждый пользователь должен иметь личный кабинет, где он может видеть список предстоящих тренингов, на которые он зарегистрирован, а также архив пройденных мероприятий. Создатели в своем кабинете дополнительно видят список созданных ими тренингов и их участников.
- Система уведомлений. Система должна автоматически отправлять пользователям уведомления о подтверждении регистрации.

Требования к структуре и функционированию сервиса.

- Управление доступом и контентом: Поскольку в системе отсутствуют пользовательские роли и регистрация, управление доступом должно быть построено на основе уникальных ссылок.
- Создание контента: Любой посетитель сервиса может создать новый тренинг без предварительной регистрации.
- Генерация ссылок: После создания тренинга система генерирует две уникальные ссылки:
- Публичная ссылка: Для просмотра тренинга. Ей можно делиться с кем угодно.
- Хранение данных: Необходима база данных для хранения информации о созданных тренингах. Структура данных должна включать:
- Коллекции шагов: Хранение последовательности шагов для каждого тренинга.
- Данные шага: Каждый шаг должен содержать исходное изображение (скриншот), координаты и тип выделенной области, а также текст аннотации.
- Связь с URL: Сопоставление уникальных публичных и приватных ссылок с конкретным тренингом.

– Основной модуль: Конструктор тренингов: Это ключевой компонент сервиса, предоставляющий интерфейс для создания и редактирования инструкций. Он должен включать:

– Загрузку изображений: Возможность загружать изображения (например, в форматах PNG, JPG) для каждого шага.

– Инструменты выделения: Простые инструменты для выделения прямоугольной или другой формы области на загруженном изображении.

– Редактор аннотаций: Текстовое поле для ввода и форматирования поясняющего текста к каждому выделению.

– Управление шагами: Возможность добавлять новые шаги, удалять существующие и изменять их порядок в рамках одного тренинга.

– Модуль просмотра тренингов: Обеспечивает интерфейс для конечных пользователей, которые просматривают инструкцию по публичной ссылке.

– Пошаговая навигация: Простой и понятный интерфейс с кнопками «Далее» и «Назад» для перемещения между шагами инструкции.

– Визуализация: Четкое отображение текущего шага: скриншот с подсвеченной областью и рядом/поверх него — соответствующая аннотация.

– Минималистичный дизайн: Интерфейс не должен быть перегружен лишними элементами, чтобы пользователь мог сосредоточиться на создании или просмотре инструкции.

Требования к способам и средствам связи для информационного обмена между компонентами сервиса. Информационный обмен между компонентами системы осуществляется через внутренние API и сервисы, обеспечивающие взаимодействие между клиентской и серверной частью приложения.

Требования к режимам функционирования сервиса

Сервис должен обеспечивать доступность и надёжность работы в любое время суток, а также иметь механизмы восстановления в случае сбоев:

- Основной режим: круглосуточная работа (24/7), обеспечивающая доступность сервиса для студентов в любое время.

Требования к эргономике и технической эстетике

- Интуитивно понятный и удобный интерфейс.
- Соответствие современным стандартам веб-дизайна и юзабилити.

Требования по стандартизации и унификации

Соответствие общепринятым стандартам веб-разработки (HTML5, CSS3, ECMAScript). Использование унифицированных протоколов и форматов данных (JSON, XML).

2.4 Срок выполнения проекта

Проект должен быть завершен 31.05.2026.

2.5 Требования к аппаратному обеспечению

Для просмотра сервиса необходимо использовать персональный компьютер или мобильное устройство с доступом в Интернет и следующими характеристиками:

- процессор Intel Core i3 или аналогичный;
- оперативная память не менее 4 Гб;
- разрешение экрана не менее 1280 x 720 пикселей.

2.6 Требования к программному обеспечению

Для просмотра сервиса необходимо использовать веб-браузер, поддерживающий HTML3 и CSS3. Рекомендуется использовать один из следующих браузеров:

- Google Chrome версии 80 или выше;
- Mozilla Firefox версии 75 или выше;
- Apple Safari версии 13 или выше;
- Microsoft Edge версии 80 или выше.

3 Руководство разработчика

Веб-сервис создан на базе актуального стека: FastAPI – современный, высоко-производительный бекенд фреймворк на языке python, VueJS и ReactJS – для фронтенд части приложения, PostgreSQL – реляционная СУБД, S3 хранилище – используется для облачного хранения загружаемых данных тренингов. Серверная часть приложения построена по паттерну стоистой архитектуры, что позволяет создавать гибкие и масштаблируемые веб-приложения.

Общая структура проекта показана на рисунке 3.1.

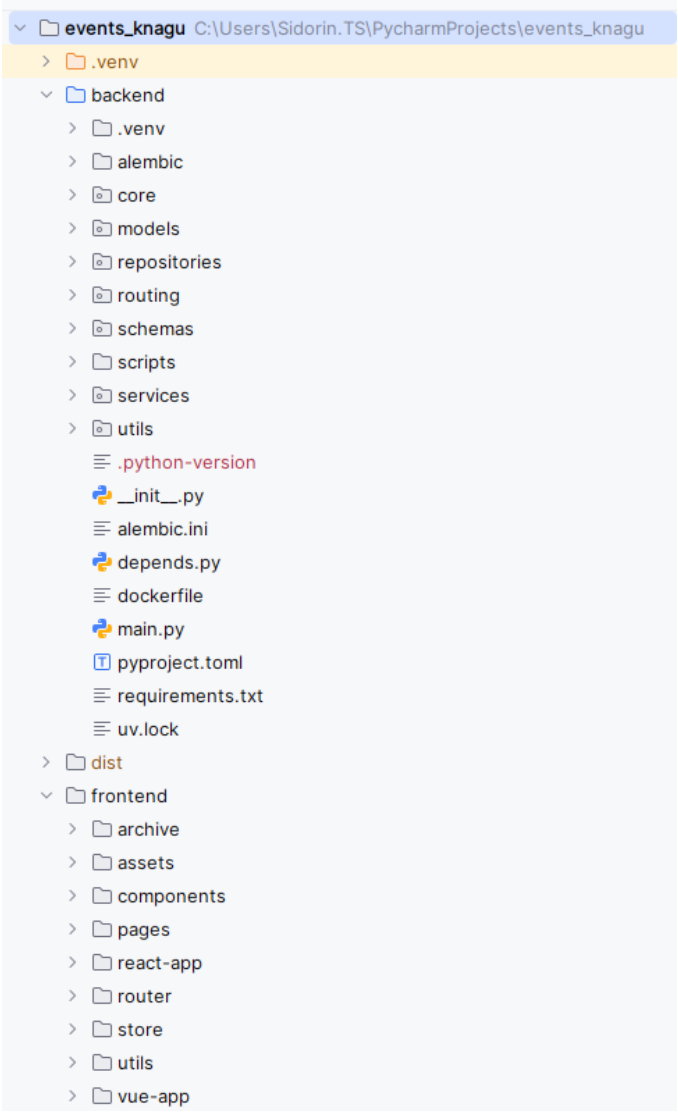

Рисунок 3.1 – Дерево проекта

Основные блоки кода показаны в листингах 3.1 – 3.3.

Листинг 3.1 – API модуль для взаимодействия с тренингами training_router.py

```
from typing import List

from fastapi import APIRouter, Depends, HTTPException, File, Up-
loadFile, status
from fastapi.security import OAuth2PasswordBearer
from pydantic import UUID4

from depends import get_trainings_service, get_s3_service,
get_user_service
from starlette import status
from services.trainings_service import TrainingsService
from services.external_services.s3_service import S3Service
from schemas.trainings import TrainingResponse, TrainingUpdate,
TrainingCreate
from services.user_service import UserService



router = APIRouter(prefix="/training", tags=["Тренинги"])
oauth2_scheme = OAuth2PasswordBearer(tokenUrl="auth/login")

@router.post("/create_training", name="Создание тренинга")
async def create_training(
    ser_data: TrainingCreate,
    token: str = Depends(oauth2_scheme),
    training_service: TrainingsService = De-
pends(get_trainings_service),
    user_service: UserService = Depends(get_user_service),
):
    """Создание нового тренинга"""
    creator_id = await user_service.get_current_user(token)
    created_training = await train-
ing_service.create_training(ser_data, creator_id.id)

    if created_training:
        return {"status": status.HTTP_201_CREATED,
                "data": created_training.dict()}
    else:
        raise HTTPException(status_code=404, detail="Тренинг не
создан")




@router.get("/{training_uuid}", response_model=TrainingResponse,
name="Получение конкретного тренинга")
```

```
async def get_training(training_uuid: UUID4, service: Train-
ingsService = Depends(get_trainings_service)):

    training = await service.get_training(training_uuid)
    if not training:
        raise HTTPException(status_code=404, detail="Тренинг не
найден")
    return training



@router.get("/my_trainings/", re-
sponse_model=list[TrainingResponse],name="Получение всех
тренингов пользователя")
async def get_my_trainings(token: str = Depends(oauth2_scheme),
                           user_service: UserService = De-
pends(get_user_service),
                           training_service: TrainingsService =
Depends(get_trainings_service)):
    user = await user_service.get_current_user(token)
    trainings = await train-
ing_service.get_trainings_by_user_id(user.id)
    if not trainings:
        raise HTTPException(status_code=404, detail="Тренинги не
найдены")
    return trainings


@router.patch("/{training_uuid}", name="Частичное обновление
тренинга")
async def patch_training(
        training_uuid: UUID4,
        training_data: TrainingUpdate,
        token: str = Depends(oauth2_scheme),
        service: TrainingsService = De-
pends(get_trainings_service),
        user_service: UserService = Depends(get_user_service),
):
    """Частичное обновление тренинга и/или его шагов"""
    user = await user_service.get_current_user(token)

    # Обновление тренинга
    updated_training = await ser-
vice.patch_training(training_uuid, training_data)

    return {
        "status": "success",
        "detail": "Данные тренинга успешно обновлены",
        "data": updated_training
    }
```

```
@router.delete("/{training_uuid}", name="Удаление тренинга по
uuid")
async def delete_training(
    training_uuid: UUID4, service: TrainingsService = De-
pends(get_trainings_service)
):
    if await service.delete_training(training_uuid):
        return {"detail": "Тренинг успешно удален"}
    else:
        raise HTTPException(status_code=404, detail="Тренинг не
найден")



@router.post("/upload-photos/{training_uuid}", name="Загрузка
фото")
async def upload_photos_by_training(
        training_uuid: UUID4,
        files: List[UploadFile] = File(..., descrip-
tion="Загрузка фото"),
        s3_service: S3Service = Depends(get_s3_service),
        trainings_service: TrainingsService = De-
pends(get_trainings_service),
        token: str = Depends(oauth2_scheme),
):
    if not files:
        raise HTTPException(status_code=400, detail="Файлы не
предоставлены")

    uploaded_urls = []

    for file in files:
        try:
            object_name =
s3_service.generate_unique_filename(file.filename)
            file_content = await file.read()
            file_url = await
s3_service.upload_file(file_content, object_name, training_uuid)
            uploaded_urls.append(file_url)
        except Exception as e:
            raise HTTPException(
                status_code=500, detail=f"Ошибка загрузки файла
{file.filename}: {str(e)}"
            )
        finally:
            await file.close()

    created_steps = await train-
ings_service.create_steps_from_photos(training_uuid, upload-
ed_urls)
```

```
    return {
        "success": True,
        "message": f"Загружено {len(uploaded_urls)} фотографий и
создано {len(created_steps)} шагов тренинга",
        "uploaded_urls": uploaded_urls,
        "created_steps": created_steps
    }



@router.delete("/delete-photo/{url_photo:path}", name="Удаление
фото по url")
async def delete_photo_by_url(url_photo: str, s3_service:
S3Service = Depends(get_s3_service)):
    if await s3_service.delete_file(url_photo):
        return {"status": "OK"}
    else:
        raise HTTPException(status_code=404, detail="Фото не
найдено")
```

Листинг 3.2 – Модуль сервиса загрузки на S3 хранилище S3_service.py

```
class S3Service:
    def __init__(
        self,
        session: AsyncSession,
        aws_access_key_id: str = configs.AWS_ACCESS_KEY_ID,
        aws_secret_access_key:         str         =         con-
figs.AWS_SECRET_ACCESS_KEY,
        region_name: str = configs.S3_REGION_NAME,
        bucket_name: str = configs.S3_BUCKET_NAME,
        endpoint_url: str = configs.S3_ENDPOINT_URL,


    ):

        self.s3_client = boto3.client(
            "s3",
            aws_access_key_id=aws_access_key_id,
            aws_secret_access_key=aws_secret_access_key,
            region_name=region_name,
            endpoint_url=endpoint_url,
        )
        self.bucket_name = bucket_name
        self.endpoint_url = endpoint_url

    def generate_unique_filename(self, original_filename: str) -
> str:
        """Генерирует уникальное имя файла на основе оригиналь-
ного имени и временной метки"""
```

```
        extension = original_filename.split(".")[-1] if "." in
original_filename else ""
        unique_id = str(uuid.uuid4())
        return f"photos/{unique_id}.{extension}"

    async def upload_file(self, file_content: bytes, ob-
ject_name: str, training_uuid: UUID4) -> str:
        """Загружает файл в S3, сохраняет в БД и возвращает
URL"""
        content_type, _ = mimetypes.guess_type(object_name)
        if content_type is None:
            content_type = "application/octet-stream"

        try:
            sha256_hash = hashlib.sha256(file_content).digest()
            sha256_hash_b64                                    =
base64.b64encode(sha256_hash).decode("utf-8")
            self.s3_client.put_object(
                Bucket=self.bucket_name,
                Key=object_name,
                Body=file_content,
                ContentType=content_type,
                ChecksumSHA256=sha256_hash_b64,
            )
            file_url = (

f"{self.endpoint_url.rstrip('/')}/{self.bucket_name}/{object_nam
e}"
            )

            return file_url
        except Exception as e:
            print(f"Ошибка при загрузке файла: {str(e)}")
            raise e




    async def delete_file(self, object_name: str):
        key      =      "/".join(object_name.split("/photos/")[-
1].split("/"))
        object_name = f"photos/{key}"
        print(f"Attempting to delete object: {object_name}")
        try:
            response = self.s3_client.delete_objects(
                Bucket=self.bucket_name,
                Delete={'Objects': [{'Key': object_name}]}
            )
            if 'Deleted' in response:
                return True
            else:
                raise              HTTPException(status_code=404,
```

```
detail="Файл не удалён")
        except Exception as e:
            raise  HTTPException(status_code=500,  detail=f"Ошибка
при удалении файла: {str(e)}")
```

Листинг 3.3 – Модуль сервиса тренингов training_service.py

```
class TrainingsService:
    def __init__(self, session: AsyncSession):
        self.repo = TrainingRepository(session)
        self.session = session

    async def create_training(
            self, training_data: TrainingCreate, creator_id: int
    ):
        training_dict = training_data.model_dump(
            exclude={"steps"}
        )
        training      =      Training(**training_dict,      crea-
tor_id=creator_id)
        created_training = await self.repo.create(training)
        if not created_training:
            return False

        for step_data in training_data.steps:
            step = await self.create_training_step(step_data)
            step.training_uuid = created_training.uuid
            await self.repo.create_training_step(step)




        created_training                  =                  await
self.repo.get_by_uuid(created_training.uuid)

        if created_training:
            return                                          Train-
ingResponse.model_validate(created_training)
        return False

    async def create_training_step(
            self, step_data: TrainingStepCreate
    ) -> TrainingStep:
        action_type                    =                    await
self.repo.get_action_type(step_data.action_type_id)
        if not action_type:
            raise HTTPException(
                status_code=status.HTTP_404_NOT_FOUND,
                detail=f"Action        type        with        ID
{step_data.action_type_id} not found"
            )
```

```
        step_dict  =  step_data.model_dump(exclude={"image_uuid",
"action_type_id"})

        return TrainingStep(
            **step_dict,
            action_type=action_type,
            image_uuid=step_data.image_uuid                    if
step_data.image_uuid else None
        )

    async def get_training(self, training_uuid: UUID4) -> Train-
ingResponse:
        training = await self.repo.get_by_uuid(training_uuid)
        if not training:
            raise HTTPException(
                status_code=status.HTTP_404_NOT_FOUND,
                detail="Training not found"
            )
        return TrainingResponse.model_validate(training)

    async def get_trainings_by_params(
            self, skip: int = 0, limit: int = 100
    ) -> List[TrainingResponse]:
        trainings = await self.repo.get_all(skip, limit)
        return  [TrainingResponse.model_validate(t)  for  t  in
trainings]

    async def get_trainings_by_user_id(
            self, user_id: int
    ) -> List[TrainingResponse]:
        trainings = await self.repo.get_by_user_id(user_id)
        return   [TrainingResponse.model_validate(training)   for
training in trainings]

    async def patch_training(
            self,  training_uuid:  UUID4,  training_data:  Train-
ingUpdate
    ) -> TrainingResponse:
        """Частичное обновление тренинга и его шагов"""
        try:
            existing_training               =               await
self.repo.get_by_uuid(training_uuid)
            if not existing_training:
                raise HTTPException(
                    status_code=status.HTTP_404_NOT_FOUND,
                    detail="Training not found"
                )

            update_data                   =                   train-
ing_data.model_dump(exclude_unset=True, exclude={"steps"})
            if update_data:
```

```
                await
self.repo.patch_training_fields(training_uuid, update_data)

            if   hasattr(training_data,   'steps')   and   train-
ing_data.steps is not None:
                await    self.patch_training_steps(training_uuid,
training_data.steps)

            updated_training                =                 await
self.repo.get_by_uuid(training_uuid)
            return                                          Train-
ingResponse.model_validate(updated_training)

        except Exception as e:
            await self.session.rollback()
            raise HTTPException(
                sta-
tus_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
                detail=f"Unexpected error: {str(e)}"
            )

    async def patch_training_steps(self, training_uuid: UUID4,
                                   steps_data:
List[Union[TrainingStepCreate, TrainingStepUpdate]]):
        """Частичное обновление шагов тренинга"""
        existing_steps                  =                  await
self.repo.get_training_steps(training_uuid)
        existing_steps_dict = {step.id: step for step in exist-
ing_steps}

        for step_data in steps_data:
            # Проверяем, есть ли у шага ID
            step_id = getattr(step_data, 'id', None)

            if step_id and step_id in existing_steps_dict:
                update_data                                     =
step_data.model_dump(exclude_unset=True, exclude={"id"})
                if update_data:   # Проверяем, что есть данные
для обновления
                    await
self.repo.update_training_step(step_id, update_data)
            elif not step_id:
                step_dict                                       =
step_data.model_dump(exclude_unset=True)
                # Убедитесь, что training_uuid установлен
                step_dict['training_uuid'] = training_uuid
                new_step = TrainingStep(**step_dict)
                await self.repo.create_training_step(new_step)


    async  def  delete_training(self,  training_uuid:  UUID4)  ->
bool:
```

```
        success = await self.repo.delete(training_uuid)
        if not success:
            raise HTTPException(
                status_code=status.HTTP_404_NOT_FOUND,
                detail="Training not found"
            )
        return True


    async def get_training_steps(self, training_uuid: UUID4) ->
List[TrainingStep]:
        return await self.repo.get_training_steps(training_uuid)



    async   def   create_steps_from_photos(self,   training_uuid:
UUID4, photo_urls: List[str]) -> List[Dict]:
        training_exists                    =                    await
self.repo.check_training_exists(training_uuid)
        if not training_exists:
            raise HTTPException(
                status_code=status.HTTP_404_NOT_FOUND,
                detail=f"Training  with  UUID  {training_uuid}  not
found"
            )
        return                                                    await
self.repo.create_steps_from_photos(training_uuid, photo_urls)
```

Листинг 3.4 – Файл главной страницы пользователя personal_page.vue

```
<template>
    <q-layout view="lHh lpR fFf">
        <!-- Вверхняя часть -->
        <q-header bordered style="height: 10%;" class="row items-center">
            <q-toolbar>
                <UserCard class="q-ml-auto q-mr-xl"/>
            </q-toolbar>
        </q-header>
        <!-- Левая панель -->
        <q-drawer show-if-above side="left" bordered>
            <div class="group_button_layout">
                <router-link
                    class="custom-nav-link column"
                    v-for="nav in buttonLeftPanel"
                    :to="nav.url"
                    active-class="active-nav-link"
                    :key="nav.url"
                    >
                    <q-btn
                        flat
                        :key="nav.name"
                        class="q-pa-md row items-start full-width"
                    >
                        <q-icon :name="nav.icon" class="q-ml-lg"/>
                        <span class="q-ml-md">{{ nav.name }}</span>
                    </q-btn>
```

```
                </router-link>
            </div>
        </q-drawer>
        <!-- Контейнер в котором отрисовка страницы -->
        <q-page-container>
            <router-view />
        </q-page-container>
        <q-page-scroller position="bottom-right" :scroll-offset="150"
:offset="[18, 18]">
            <q-btn fab icon="keyboard_arrow_up" style="color: #5064F7;" />
        </q-page-scroller>
    </q-layout>
</template>

<script>
import UserCard from
'@components/for_pages/PersonalPage/for_header/UserCard.vue';
export default {
    name: 'Layout',
    components: {UserCard},
    data() {
        return {
            buttonLeftPanel: [
                {
                    'name': 'Главная',
                    'icon': 'home_outline',
                    'url': '/personal/home'
                },
                {
                    'name': 'Библиотека',
                    'icon': 'language',
                    'url': '/personal/library'
                },
                {
                    'name': 'Тренинги',
                    'icon': 'list_alt',
                    'url': '/personal/traning'
                },
                {
                    'name': 'Поддержка',
                    'icon': 'info',
                    'url': '/personal/help'
                }
            ]
        }
    },
}
</script>

<style scoped>
.q-header {
    background-color: white;
}

.group_button_layout {
    margin-top: 25%;
    color: #808080;
}

.custom-nav-link {
    color: #808080;
    text-decoration: none;
```

```
}

.active-nav-link {
    background-color: #FAFAFC;
    color: #5064F7;
    border-left: 5px solid #5064F7;
}
</style>
```

Листинг 3.5 – Файл редактировнаия тренинга EditPage.vue

```
import React from "react";
import styles from '@assets/styles/for-components/drop-down.module.css';
import menuIcon from '@assets/img/menu.svg';
import dndStep from '@assets/img/dnd-step.svg';
import add from '@assets/img/add.svg';
import pencil from '@assets/img/pencil.svg';
import check from '@assets/img/check.svg';

/*
        Компонент - выпадающий список
*/
function DropDown() {
        return (
                        <div className={styles['drop-down']}>
                                Назад
                        </div>
        );
}

/*
        Компонент - выпадающий список с шагами
*/
function DropDownSteps(props) {
        //стили
        let styleNameSteps = styles['drop-down-step'] + " " + 'q-ml-xl';
        let styleNameTopBar = styles['top-bar-step'] + " " + 'q-ml-xl';
        let styleStep = styles.step;

        //статус 0
        let iconCheck = <img className={styles['check-icon']} src={check}
alt="" />;
        //статус 1
        let iconStepDnd = <img className={styles['dnd-icon']} src={dndStep}
alt="" />;
        const [statusMod, setStatusMod] = React.useState(false);

        function handleMod() {
                setStatusMod(!statusMod);
        }

        const [selectedStep, setSelectedStep] = Re-
act.useState(props.props.props.props.steps[0]);

        function selectStep(step) {
                return function() {
                        setSelectedStep(step);
                }
        }
```

```
        let steps = props.props.props.props.steps.map((step) =>
                <div onClick={selectStep(step)} className={styleStep}
key={step.id}>
                        {statusMod === true ? iconStepDnd : ''}
                        {selectedStep === step ? iconCheck : ''}
                        <span>{step.id}</span>
                </div>
        );
        return (
                <>
                        <div className={styleNameTopBar}>
                                <span className="q-ml-md">Шаги</span>
                                <div className="q-ml-auto q-gutter-md">
                                        <img onClick={handleMod}
src={pencil} alt="" />
                                        <img src={add} alt="" class-
Name="q-mr-md" />
                                </div>
                        </div>
                        <div className={styleNameSteps}>
                                {steps}
                        </div>
                </>
        );
}

/*
        Компонент - группа выпадающих списков
*/
function ButtonsGroup(props) {
        const [statusDropdown, setStatusDropdown] = React.useState(false);
        const [statusDropDownStep, setStatusDropDownStep] = Re-
act.useState(false);

        return (
                <div className="fit">
                        <div className={styles['left-button']}>
                                <button
                                        className={statusDropdown ?
styles['button-active'] : ''}
                                        onClick={() => {setSta-
tusDropdown(!statusDropdown); setStatusDropDownStep(false)}}
                                >
                                        <img src={menuIcon} alt="menu" />
                                </button>
                                <button
                                        className={`${styles['button-steps']}
${statusDropDownStep ? styles['button-active'] : ''}`}
                                        onClick={() => {setStatusDropDown-
Step(!statusDropDownStep); setStatusDropdown(false)}}
                                >
                                        Страница 1
                                </button>
                        </div>
                        {statusDropdown && <DropDown />}
                        {statusDropDownStep && <DropDownSteps props={props}/>}
                </div>
        );
}

export default ButtonsGroup;
```

Листинг 3.6 – Файл запуска бекенд части main.py

```
"""
Точка входа в backend
"""
#
from fastapi import FastAPI, APIRouter
from routing import auth_router, training_router
from core.config import configs
from core.create_base_app import create_base_app


app = create_base_app(configs)
app.include_router(auth_router.router)
app.include_router(training_router.router)

# docker run -p 8001:8001 feec3ca171c7

if __name__ == "__main__":
    import uvicorn
    uvicorn.run("main:app", host=configs.HOST, port=configs.PORT, re-
load=True)

# uvicorn.run("main:app", host="localhost", port=8000, reload=True)
```

4 Руководство пользователя

Основные окна приложения показаны на рисунках 4.1 – 4.5.

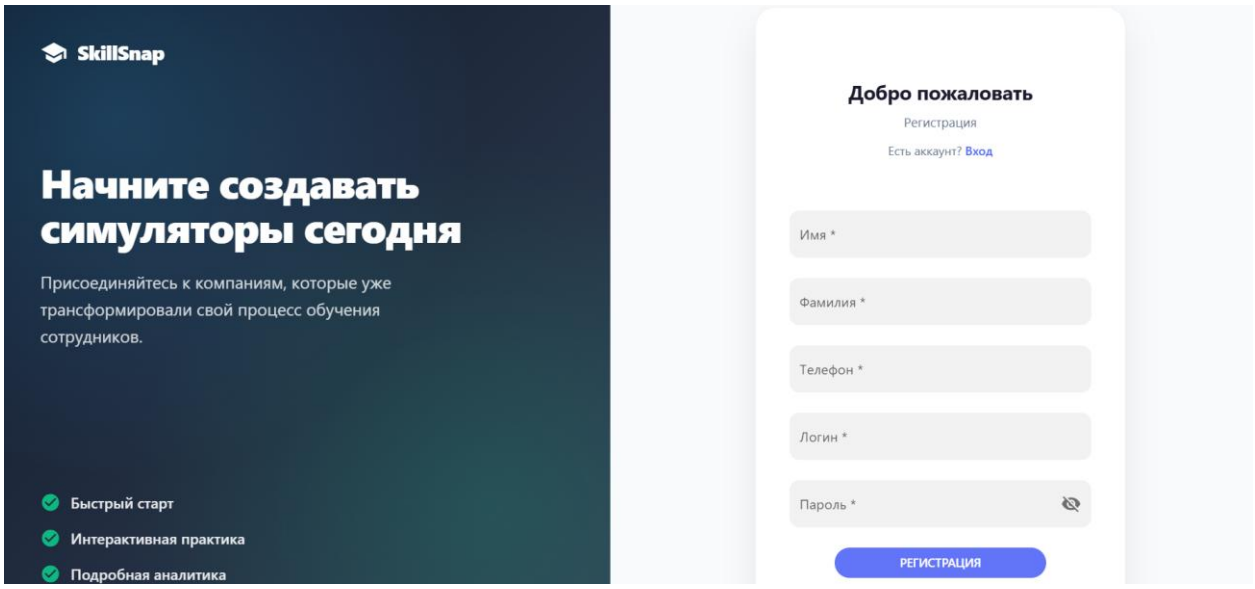

Рисунок 4.1 – Процесс регистрации на сервисе

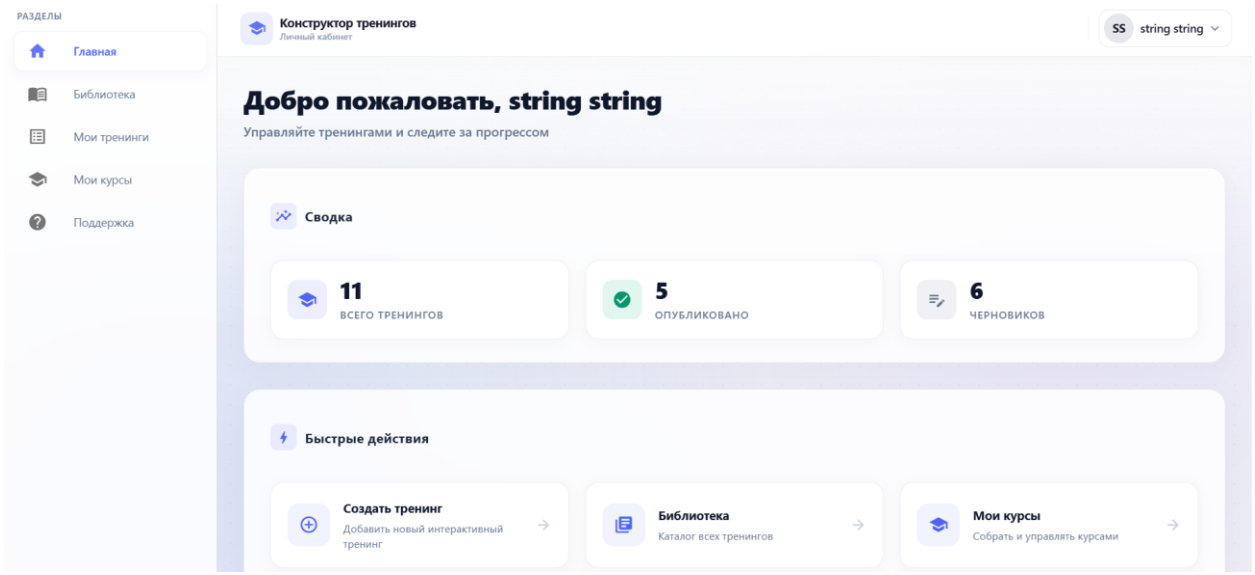

Рисунок 4.2 – Окно созданных тренингов пользователя

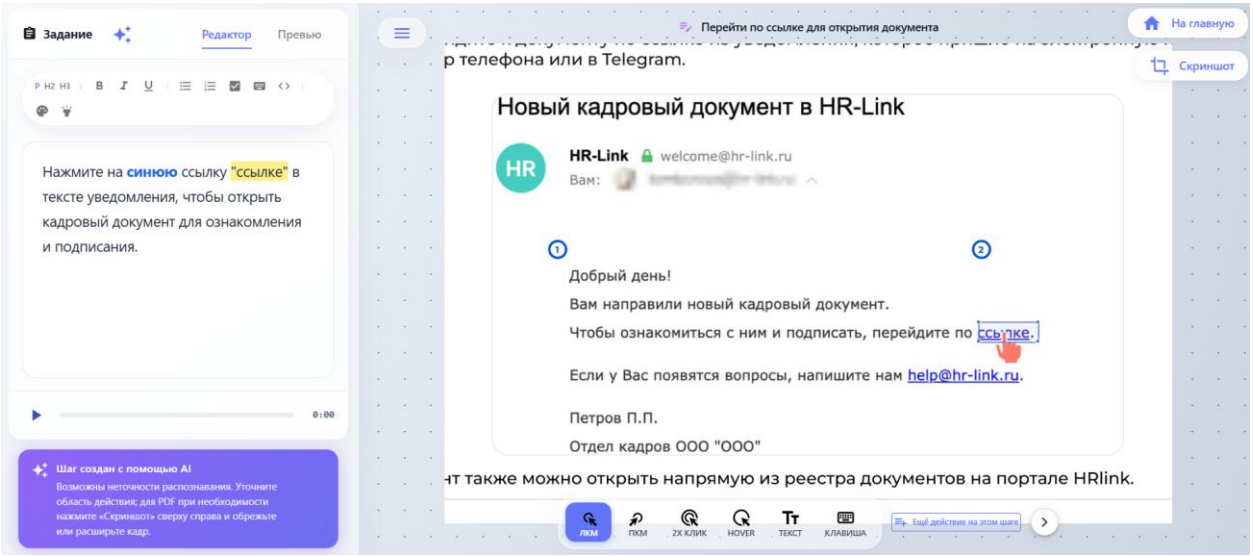

Рисунок 4.3 – Базовое создание тренинга (основные поля)

При открытии тренинга, если в нем еще нет шагов, предлагается загрузить фото, шаги тренинга создадутся автоматически.

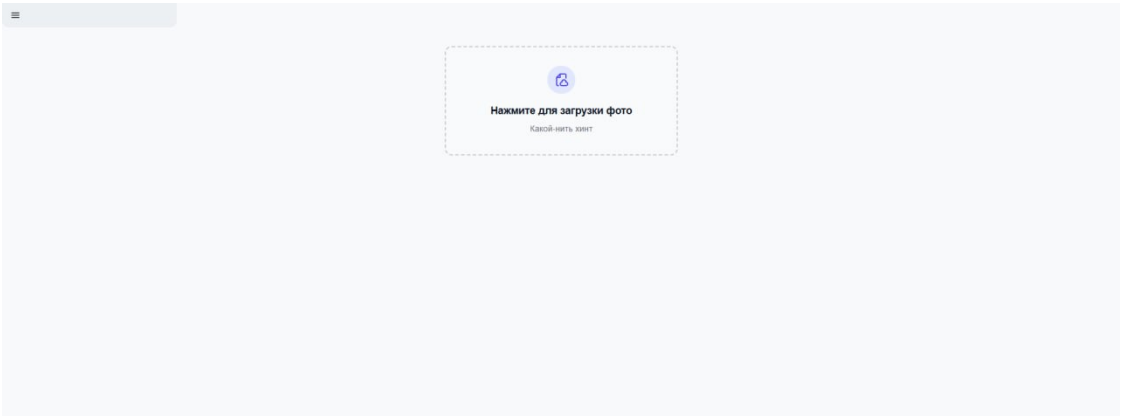

Рисунок 4.4 – Загрузка контента тренинга

После загрузки изображений есть возможность интерактивно, с помощью Drag And Drop выставить порядок шагов по желанию, а затем выделить на каждом изображении необходимые области, предварительно выбрав нужный тип выделения (клик, двойной клик, ввести текст).

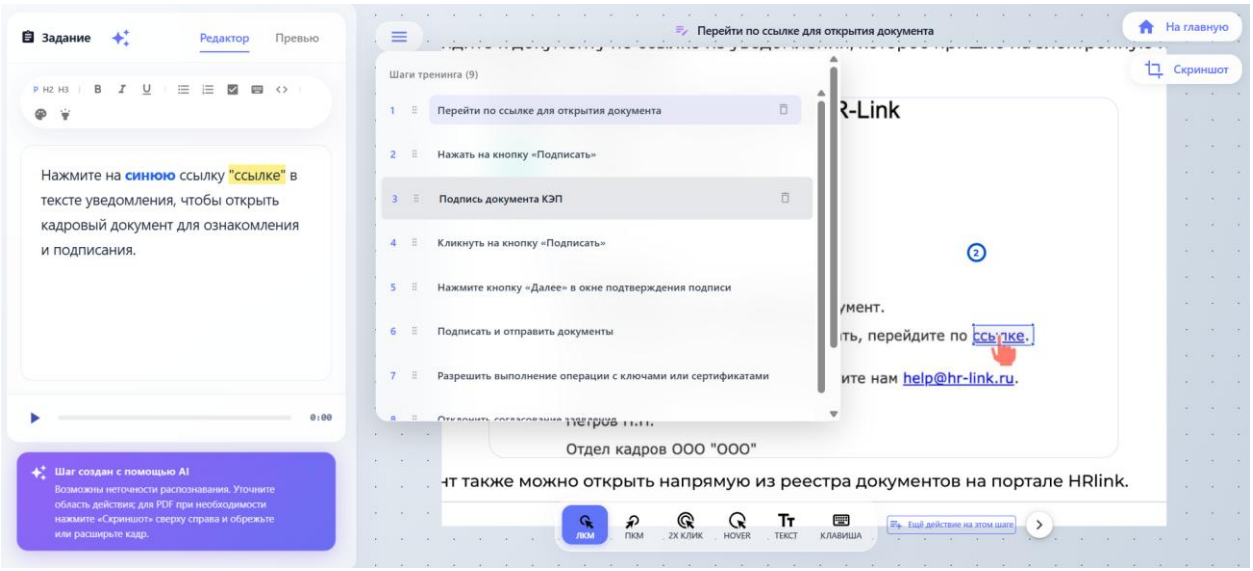

Рисунок 4.5 – Окно редактирования тренинга

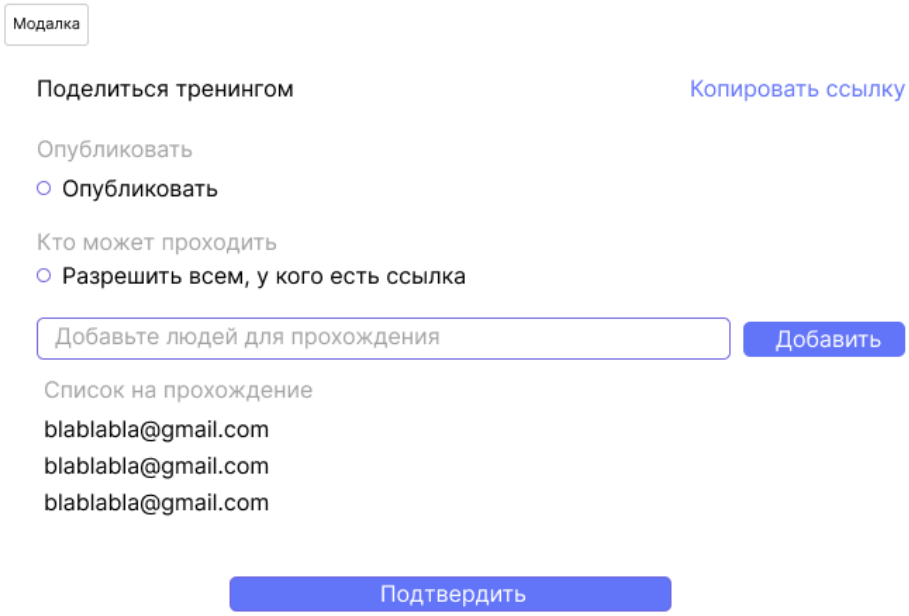

Рисунок 4.6 – экспорт тренинга

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС

_____ Е.М. Димитриади
(подпись)

«____ » _____ 20___ г.

Декан ФКТ

_____И.А. Трещев
(подпись)

УТВЕРЖДАЮ

И.о. проректора
по научной работе

_____А.В. Космынин
(подпись)

«____ » _____ 20___ г.

АКТ
о приеме проекта
Разработка веб-сервиса создания электронных тренингов

г. Комсомольск-на-Амуре « » _____ 20__ г.

Комиссия в составе представителей:

со стороны заказчика

- Е.Э. Шаповалов – руководителя СПБ «DeCode»,
- И.А. Трещев – декана _ФКТ__

со стороны исполнителя

- Е.Б. Абарникова – руководителя проекта,
- Т.С. Сидорин – 2ИТб-1
- К.М. Черненко – 2ИТб-1

составила акт о нижеследующем:

«Исполнитель» передает проект Проектирование веб-сервиса создания электронных тренингов, в составе:

1 Руководство пользователя

2 Руководство разработчика

Руководитель проекта _____ Е.Б. Абарникова
(подпись, дата)

Исполнители проекта _____ Т.С. Сидорин
(подпись, дата)

_____ К.М. Черненко
(подпись, дата)