

**ОТЧЕТ**

по производственной практике ПП.03.01

по профессиональному модулю ПМ.03 **«Участие в интеграции программных модулей»**

по специальности 09.02.03 «Программирование в компьютерных системах»

Выполнил студент гр. П1-18

Герасимов Дмитрий

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись)

Проверил преподаватель

Бобкова Н. Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2021

**Оглавление**

[Введение 3](#_Toc94251049)

[1. Общие сведения о предприятии 4](#_Toc94251050)

[1.1. Сфера деятельности предприятия, история появления и развития предприятия 4](#_Toc94251051)

[1.2. Структура предприятия, функции подразделения и взаимосвязь с другими подразделениями 4](#_Toc94251052)

[1.3. Список задач подразделения в сфере информационных технологий. 6](#_Toc94251053)

[2. Содержание выполняемых видов работ 7](#_Toc94251054)

[2.1 Анализ существующих в подразделении локальных и распределенных баз данных, приложений баз данных 7](#_Toc94251055)

[2.2 Получение задания на конкретную разработку 7](#_Toc94251056)

[2.3 Изучение документооборота, подлежащего автоматизации 7](#_Toc94251057)

[2.4 Построение IDEF-модели 8](#_Toc94251058)

[2.6 Разработка модулей программного средства 10](#_Toc94251059)

[2.7 Разработка пользовательского интерфейса программного средства 36](#_Toc94251060)

[2.8 Отладка и тестирование программного средства 37](#_Toc94251061)

[Заключение 39](#_Toc94251062)

[Список литературы и интернет-ресурсов 40](#_Toc94251063)

[Дневник практики 41](#_Toc94251064)

[Приложение 1. Техническое задание 42](#_Toc94251065)

[Приложение 2. Руководство оператора 48](#_Toc94251066)

[Приложение 3. Руководство программиста 50](#_Toc94251067)

[Приложение 4. Листинг кода 51](#_Toc94251070)

# Введение

Студентом группы П1-18 Герасимовым Дмитрием была пройдена практика по модулю ПМ.03 «Разработка программных модулей программного обеспечения для компьютерных систем» с целью получения практических навыков по разработке программных модулей и мобильных приложений.

# Общие сведения о предприятии

## Сфера деятельности предприятия, история появления и развития предприятия

Данное предприятие работает в сфере образования. Университет образован 16 июля 1998 года в форме некоммерческой организации с названием: Негосударственное образовательное учреждение «Королевская академия управления, экономики и социологии».

Технологический университет (ранее Финансово-технологическая академия; Королевский институт управления, экономики и социологии) создан для подготовки кадров новой информации, воспроизводства интеллектуальных ресурсов, формирования инновационных проектов и технологий. Академия находится в наукограде Королеве Московской области – уникальном центре интеллектуальных ресурсов, которые используются для интеграции важнейших знаний и создания систем глобального масштаба.

20 января 2015 года постановлением Правительства Московской области Академии присвоен статус «университета» и вуз переименован в Государственное бюджетное образовательное учреждение высшего образования Московской области «Технологический университет».

## Структура предприятия, функции подразделения и взаимосвязь с другими подразделениями

Ниже представлена организационная структура предприятия:

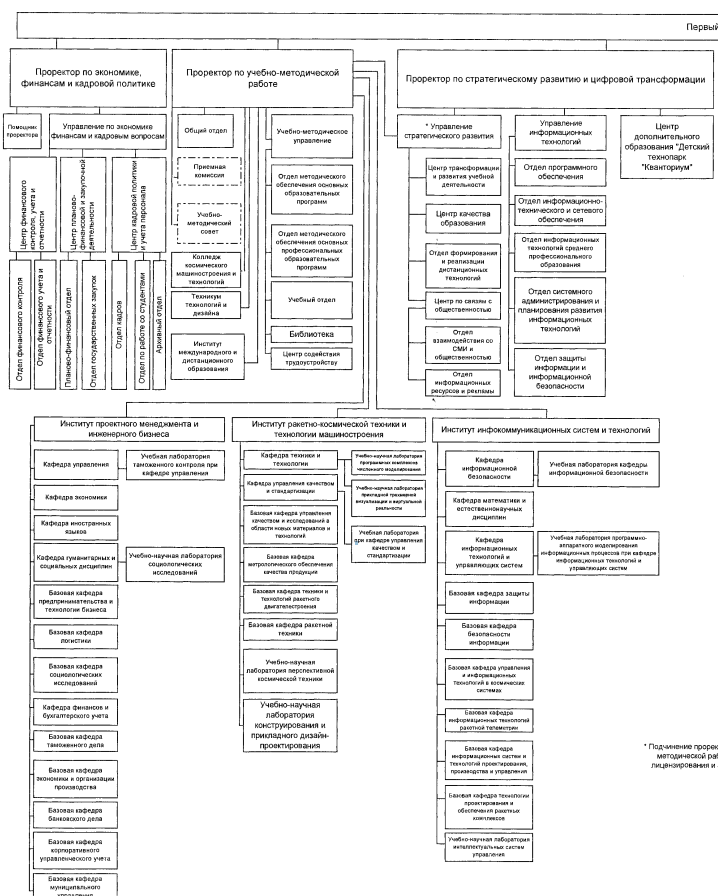


Рисунок 1. Организационная структура предприятия. Часть 1



Рисунок 2. Организационная структура предприятия. Часть 2

## Список задач подразделения в сфере информационных технологий.

Отдел формирования и реализации дистанционного обучения обеспечивает:

* внедрение в учебный процесс дистанционных образовательных технологий;
* организует учебный процесс дистанционных образовательных технологий;
* формирование и развитие базы электронных образовательных ресурсов системы дистанционного обучения;
* разработка и реализация программ обучения участников образовательного процесса.

# Содержание выполняемых видов работ

## Анализ существующих в подразделении локальных и распределенных баз данных, приложений баз данных

На данный момент в подразделении отсутствуют базы данных, приложения базы данных

## Получение задания на конкретную разработку

Задача состоит в проектировании и создании образовательного портала для прохождения курсов.

Необходимо разработать Web-сайт и Android приложение для прохождения курсов на образовательном портале. Курсы должны состоять из шагов, которые будут проходить пользователи. Также курс содержит задания, за выполнение которых пользователь получает баллы. Пользователь может просматривать прогресс в курсах, которые он проходит.

Инструменты для реализации серверной части: PostgreSQL [1].

Инструменты для реализации клиентской части:

* Web: Django [2] + JavaScript [3].
* Android: Kotlin [4] + Android SDK [5].

## Изучение документооборота, подлежащего автоматизации

Документооборот в отделе формирования и реализации дистанционных технологий представляет из себя заявку от обучающегося для поступления на курс (на вход), а по окончанию образовательного процесса - оценочная ведомость (на выход).



Рисунок 3. Схема документооборота

## Построение IDEF-модели

Ниже приведены IDEF-модели подразделения:

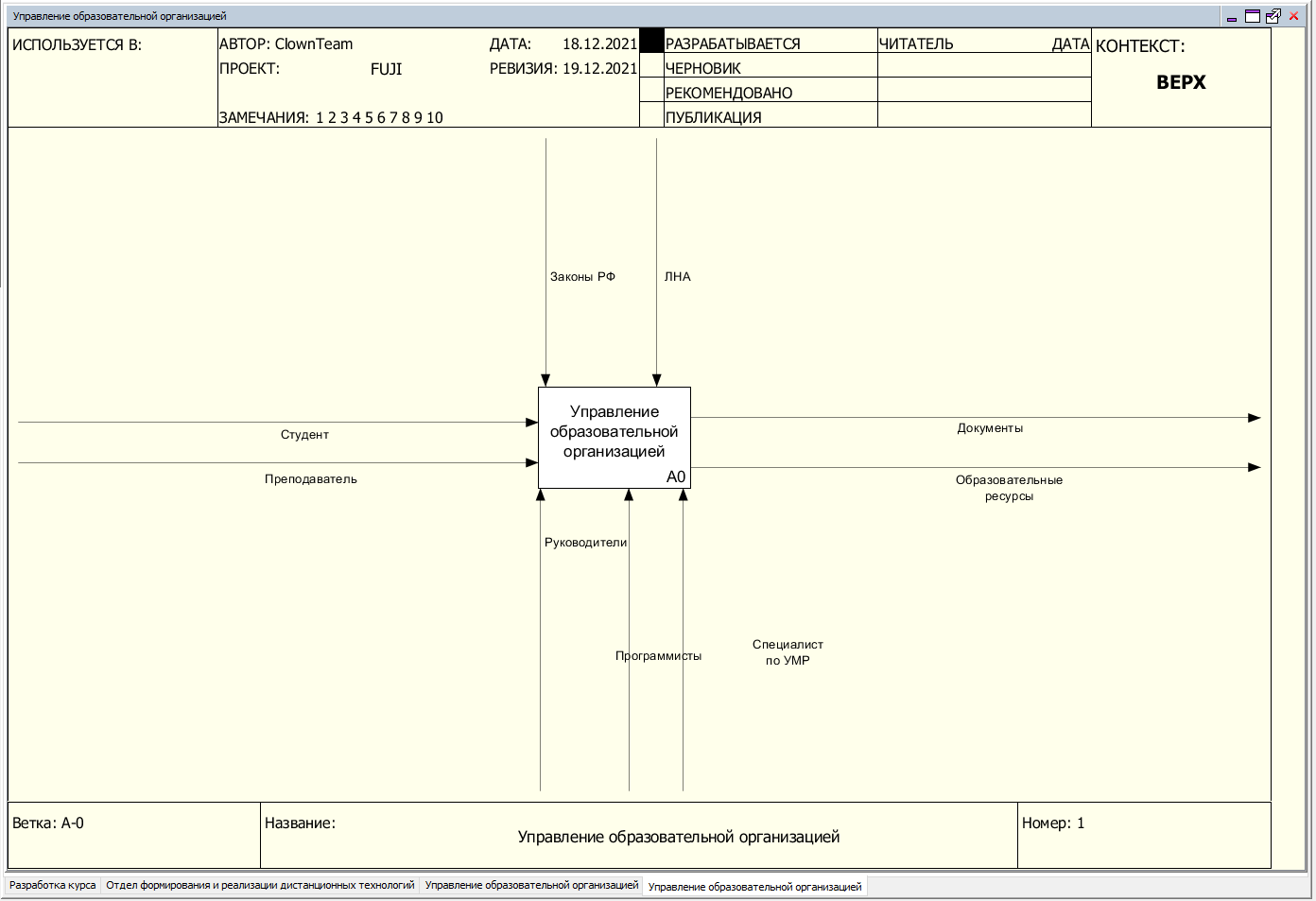


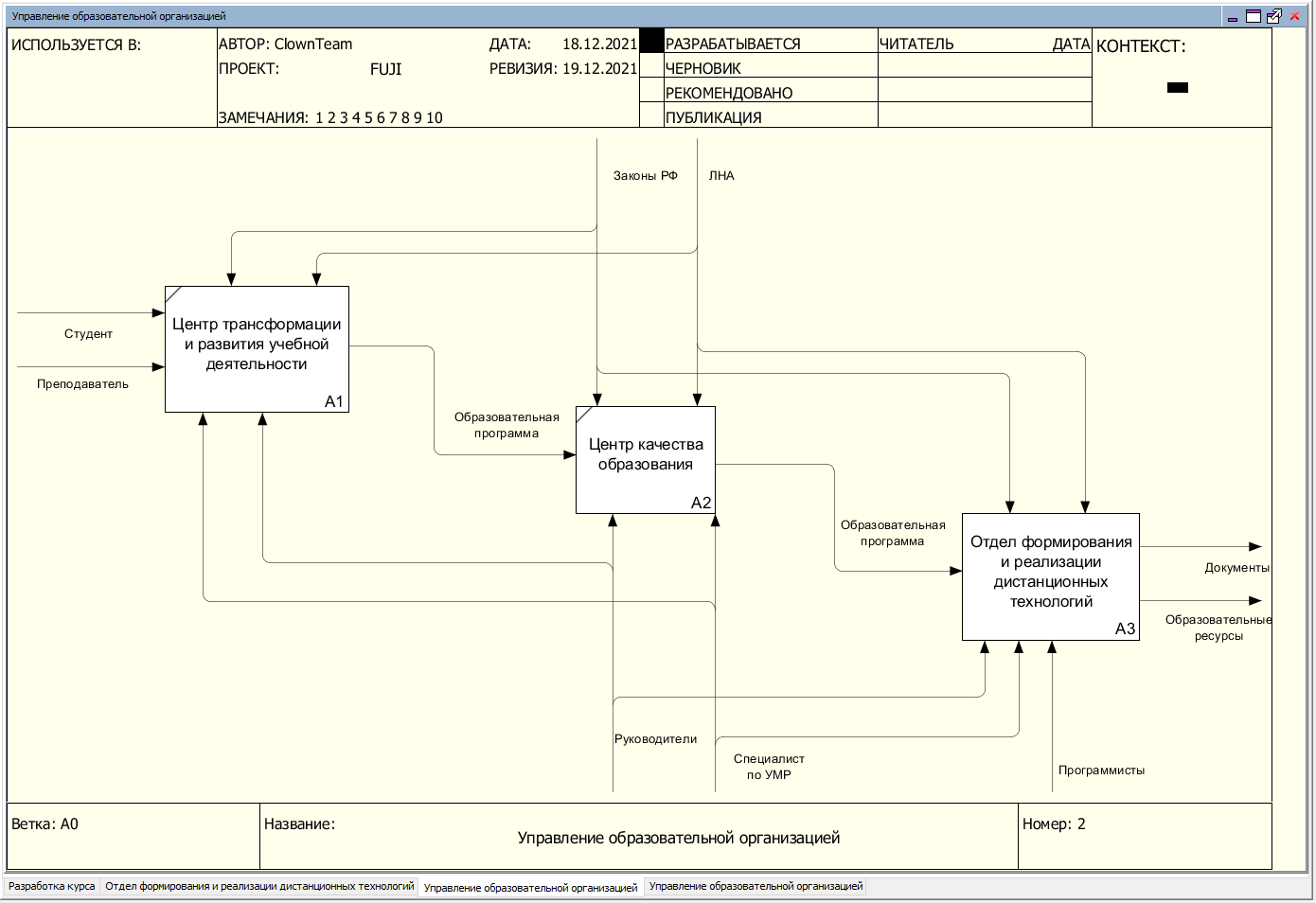
Рисунок 4. IDEF-модель 1-й уровень

Рисунок 5. IDEF-модель 2-й уровень

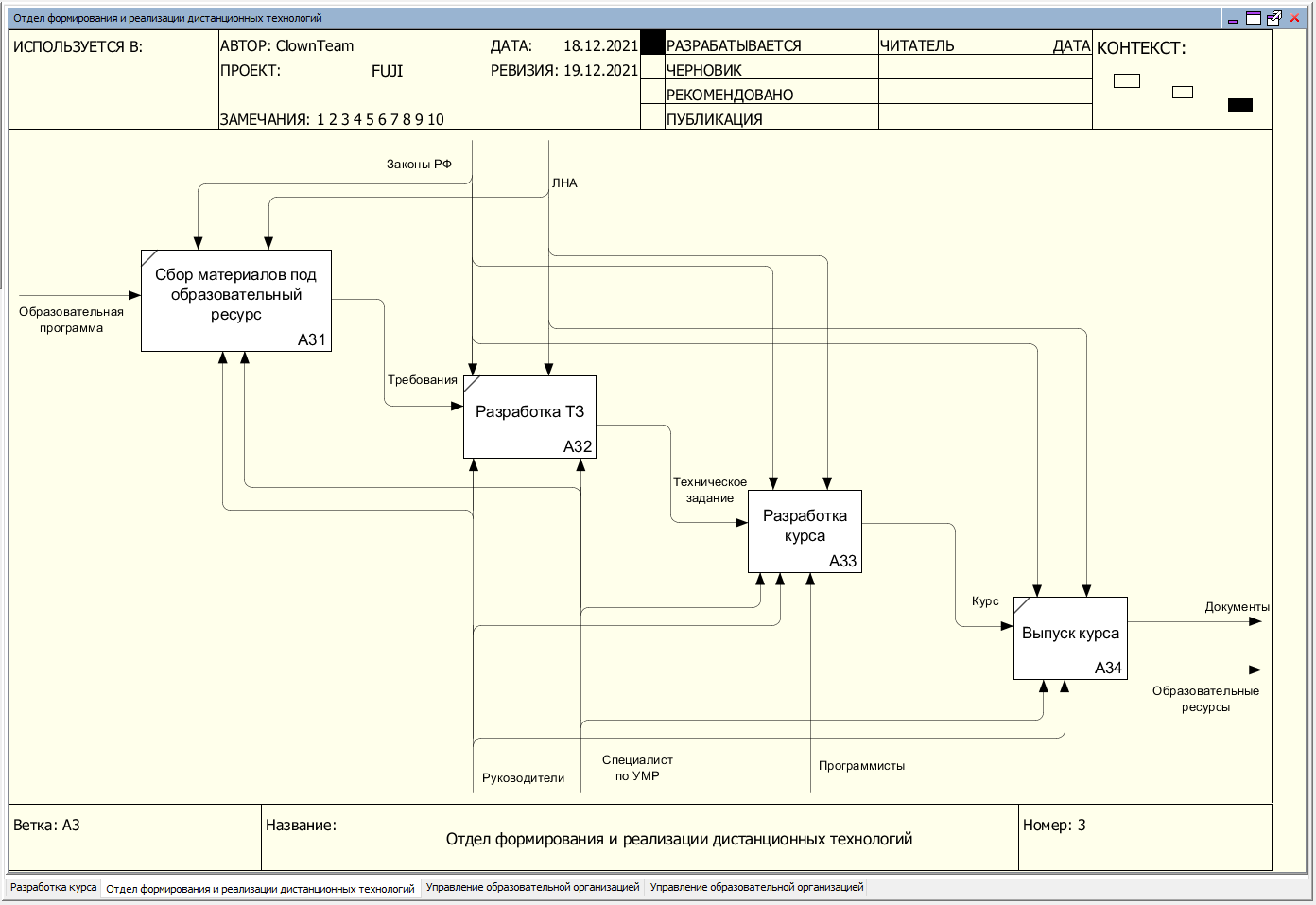


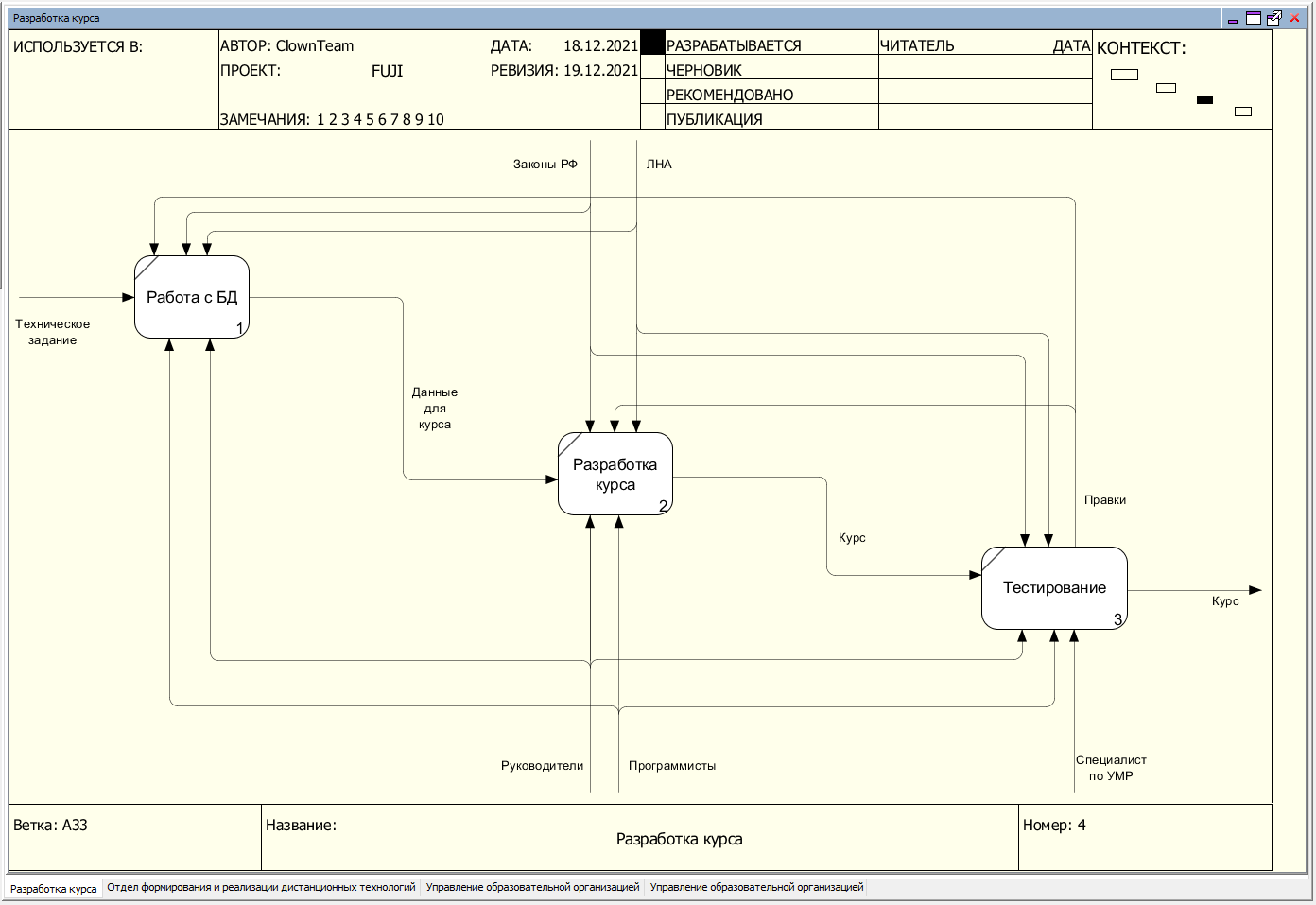
Рисунок 6. IDEF-модель 3-й уровень

Рисунок 7. IDEF-модель 4-й уровень

* 1. **Составление технического задания на разработку программного средства**

Техническое задание представлено в приложении 1.

## Разработка модулей программного средства

Были разработаны следующие модули:

1. API:
   1. urls
   2. views
   3. admin
   4. serializers
   5. models
2. Сайт
   1. Components
      1. HeaderLightBlack
      2. HeaderPurpleWhite
      3. HeaderImgWhite
      4. Footer
   2. Pages
      1. Course
      2. Index
      3. \_id
   3. \_mixins
      1. \_font-face
      2. \_header-style
      3. \_media
   4. \_style
      1. \_course
      2. \_footer
      3. \_header
      4. \_index
   5. \_fonts
   6. \_vars
   7. default

Ниже представлен листинг с функцией списка курсов из модуля ui-courseList:

**Главный модуль**

'''

<template>

<div id="main-case">

<HeaderPurpleWhite />

<div class="blocks">

<div class="base">

<div class="fixed-container">

<div class="base\_\_body">

<div class="base\_\_info">

<h2 class="base\_\_title">Стань умнее вместе с FUJI</h2>

<p class="base\_\_description">Правильно произносить «Фудзи»</p>

</div>

<div class="base\_\_icon">

<img src="img/mountain.gif" alt="Горы">

</div>

</div>

</div>

</div>

<div class="section">

<div class="fixed-container">

<div class="section\_\_body">

<div class="section\_\_header">

<h3 class="section\_\_title">Наши разделы</h3>

<div class="line"/>

</div>

<div class="section\_\_content">

<div class="section\_\_card">

<div class="item-section">

<div class="item-section\_\_text">

<h5 class="item-section\_\_title">Курсы</h5>

<p class="item-section\_\_description">Серия из лекций позволяющая изучить какой-либо предмет или обучиться навыку</p>

<div class="item-section\_\_link">

<nuxt-link to="/courses">Перейти</nuxt-link>

</div>

</div>

<div class="item-section\_\_icon">

<img src="img/book.svg" alt="Карандаш">

</div>

</div>

</div>

<div class="section\_\_card">

<div class="item-section">

<div class="item-section\_\_text">

<h5 class="item-section\_\_title">Организации</h5>

<p class="item-section\_\_description">Это люди, компании, школы, которые создают для вас весь контент (курсы, лекции)</p>

<div class="item-section\_\_link">

<nuxt-link to="#">Перейти</nuxt-link>

</div>

</div>

<div class="item-section\_\_icon">

<img src="img/pencil.svg" alt="Карандаш">

</div>

</div>

</div>

<div class="section\_\_card">

<div class="item-section">

<div class="item-section\_\_text">

<h5 class="item-section\_\_title">Лекции</h5>

<p class="item-section\_\_description">Структурированный разбор темы</p>

<div class="item-section\_\_link">

<nuxt-link to="#">Перейти</nuxt-link>

</div>

</div>

<div class="item-section\_\_icon">

<img src="img/sheet.svg" alt="Карандаш">

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<div class="mobile">

<div class="fixed-container">

<div class="mobile\_\_body">

<div class="mobile\_\_text">

<h4 class="mobile\_\_title">Изучайте курсы везде</h4>

<p class="mobile\_\_description">FUJI доступен и на Android</p>

<div class="mobile\_\_google-play">

<img src="img/google\_play.svg" alt="Доступно в Google Play">

</div>

</div>

<div class="mobile\_\_icon">

<img src="img/android.svg" alt="android">

</div>

</div>

</div>

</div>

</div>

<Footer />

</div>

</template>

<script>

import HeaderPurpleWhite from "@/components/HeaderPurpleWhite"

import Footer from "@/components/Footer";

export default {

name: 'IndexPage',

components: {HeaderPurpleWhite, Footer}

}

</script>

<style lang="scss" scoped>

@import "theme/\_style/index";

</style>

'''

**Модуль сериализации данных для курсов**

from rest\_framework import serializers

from .models import Author, Course, Profile

from .models import CourseInfo, CourseFit, CourseSkill, CourseStars, CourseComment

from django.contrib.auth.models import User

class RegisterSerializer(serializers.ModelSerializer):

re\_password = serializers.CharField(write\_only=True)

class Meta:

model = User

fields = ('id', 'username', 'password', 're\_password', 'first\_name', 'last\_name', 'email')

extra\_kwargs = {

'password': {"write\_only": True}

}

def create(self, validated\_data):

username = validated\_data['username']

password = validated\_data['password']

re\_password = validated\_data['re\_password']

first\_name = validated\_data['first\_name']

last\_name = validated\_data['last\_name']

email = validated\_data['email']

if password != re\_password:

raise serializers.ValidationError({'password': "Пароли не совпадают"})

user = User(

username=username,

first\_name=first\_name,

last\_name=last\_name,

email=email

)

user.set\_password(password)

user.save()

profile = Profile(user=user)

profile.save()

return user

class ProfileSerializer(serializers.ModelSerializer):

class Meta:

model = Profile

fields = '\_\_all\_\_'

class UserSerializer(serializers.ModelSerializer):

profile = serializers.SerializerMethodField()

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Profile(models.Model):

"""

Пользователь

"""

# Расширение класса User (login, password, first\_name, last\_name)

user = models.OneToOneField(User, on\_delete=models.CASCADE)

avatar\_url = models.ImageField(default="default.jpg")

about\_me\_text = models.TextField(blank=True, default="")

def \_\_str\_\_(self):

return f'{self.user}'

class Author(models.Model):

"""

Автор курса

"""

name = models.CharField(max\_length=50)

def \_\_str\_\_(self):

return self.name

class Course(models.Model):

"""

Курс

"""

title = models.CharField(max\_length=100)

description = models.TextField()

price = models.IntegerField()

author = models.ForeignKey(Author, on\_delete=models.CASCADE)

image = models.ImageField()

duration\_in\_minutes = models.IntegerField()

rating = models.FloatField(default=0)

members\_amount = models.IntegerField(default=0)

has\_certificate = models.BooleanField()

max\_progress\_points = models.IntegerField()

def \_\_str\_\_(self):

return self.title

class CourseInfo(models.Model):

"""

Страница курса

"""

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

goal\_description = models.TextField()

def \_\_str\_\_(self):

return self.course.title

class CourseFit(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

title = models.CharField(max\_length=100)

description = models.TextField()

def \_\_str\_\_(self):

return f'{self.course\_info.course.title}: ({self.title})'

class CourseSkill(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

name = models.CharField(max\_length=255)

def \_\_str\_\_(self):

return f'{self.course\_info.course.title}: ({self.name})'

class CourseStars(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

five\_stars\_count = models.IntegerField(default=0)

four\_stars\_count = models.IntegerField(default=0)

three\_stars\_count = models.IntegerField(default=0)

two\_stars\_count = models.IntegerField(default=0)

one\_stars\_count = models.IntegerField(default=0)

def \_\_str\_\_(self):

return self.course\_info.course.title

class CourseComment(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

created\_at = models.DateTimeField(auto\_now\_add=True)

comment = models.TextField()

stars\_count = models.IntegerField()

def \_\_str\_\_(self):

return f'{self.course\_info.course.title}: ({self.user.user.username})'

class Module(models.Model):

"""

Модуль курса

"""

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

title = models.CharField(max\_length=150)

def \_\_str\_\_(self):

return self.title

class Lesson(models.Model):

"""

Урок модуля к курсу

"""

module = models.ForeignKey(Module, on\_delete=models.CASCADE)

title = models.CharField(max\_length=150)

def \_\_str\_\_(self):

return self.title

class Step(models.Model):

"""

Степ к уроку к модулю к курсу

"""

lesson = models.ForeignKey(Lesson, on\_delete=models.CASCADE)

title = models.CharField(max\_length=150)

content = models.TextField()

def \_\_str\_\_(self):

return self.title

class UserToCourse(models.Model):

"""

Пользователь к курсу. Какие курсы проходит пользователь

"""

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

progress\_points = models.IntegerField()

class UserToModule(models.Model):

"""

Пользователь к Модулю. Какой модуль в курсе проходит пользователь

"""

module = models.ForeignKey(Module, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

class UserToLesson(models.Model):

"""

Пользователь к Уроку. Какой урок в курсе проходит пользователь

"""

lesson = models.ForeignKey(Lesson, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

class UserToStep(models.Model):

"""

Пользователь к Степу. Какой степ в курсе проходит пользователь

"""

step = models.ForeignKey(Step, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

class Meta:

model = User

fields = '\_\_all\_\_'

def get\_profile(self, user):

return ProfileSerializer(user.profile).data

class CourseSerializer(serializers.ModelSerializer):

author = serializers.SlugRelatedField(slug\_field="name", queryset=Author.objects.all())

class Meta:

model = Course

fields = '\_\_all\_\_'

def get\_info(self, pk):

queryset = CourseInfo.objects.get(course\_id=pk)

return CourseInfoSerializer(queryset).data

class CourseInfoSerializer(serializers.ModelSerializer):

fit = serializers.SerializerMethodField()

skill = serializers.SerializerMethodField()

stars = serializers.SerializerMethodField()

comments = serializers.SerializerMethodField()

class Meta:

model = CourseInfo

fields = ['goal\_description', 'fit', 'skill', 'stars', 'comments']

def get\_fit(self, course\_info):

queryset = CourseFit.objects.filter(course\_info\_id=course\_info.pk)

return CourseFitSerializer(queryset, many=True).data

def get\_skill(self, course\_info):

queryset = CourseSkill.objects.filter(course\_info\_id=course\_info.pk)

return CourseSkillSerializer(queryset, many=True).data

def get\_stars(self, course\_info):

queryset = CourseStars.objects.get(course\_info\_id=course\_info.pk)

return CourseStarsSerializer(queryset).data

def get\_comments(self, course\_info):

queryset = CourseComment.objects.filter(course\_info\_id=course\_info.pk)

return CourseCommentSerializer(queryset, many=True).data

class CourseFitSerializer(serializers.ModelSerializer):

class Meta:

model = CourseFit

fields = ['title', 'description']

# fields = '\_\_all\_\_'

class CourseSkillSerializer(serializers.ModelSerializer):

class Meta:

model = CourseSkill

fields = ['name']

class CourseStarsSerializer(serializers.ModelSerializer):

class Meta:

model = CourseStars

fields = ['five\_stars\_count', 'four\_stars\_count', 'three\_stars\_count', 'two\_stars\_count', 'one\_stars\_count']

class CourseCommentSerializer(serializers.ModelSerializer):

user = serializers.SerializerMethodField()

class Meta:

model = CourseComment

fields = ['user', 'created\_at', 'comment', 'stars\_count']

def get\_user(self, course\_info):

user = User.objects.get(pk=course\_info.pk)

return UserCommentSerializer(user).data

class UserCommentSerializer(serializers.ModelSerializer):

class Meta:

model = User

fields = ['id', 'username', 'first\_name', 'last\_name']

**Модуль моделей данных для курсов**

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Profile(models.Model):

"""

Пользователь

"""

# Расширение класса User (login, password, first\_name, last\_name)

user = models.OneToOneField(User, on\_delete=models.CASCADE)

avatar\_url = models.ImageField(default="default.jpg")

about\_me\_text = models.TextField(blank=True, default="")

def \_\_str\_\_(self):

return f'{self.user}'

class Author(models.Model):

"""

Автор курса

"""

name = models.CharField(max\_length=50)

def \_\_str\_\_(self):

return self.name

class Course(models.Model):

"""

Курс

"""

title = models.CharField(max\_length=100)

description = models.TextField()

price = models.IntegerField()

author = models.ForeignKey(Author, on\_delete=models.CASCADE)

image = models.ImageField()

duration\_in\_minutes = models.IntegerField()

rating = models.FloatField(default=0)

members\_amount = models.IntegerField(default=0)

has\_certificate = models.BooleanField()

max\_progress\_points = models.IntegerField()

def \_\_str\_\_(self):

return self.title

class CourseInfo(models.Model):

"""

Страница курса

"""

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

goal\_description = models.TextField()

def \_\_str\_\_(self):

return self.course.title

class CourseFit(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

title = models.CharField(max\_length=100)

description = models.TextField()

def \_\_str\_\_(self):

return f'{self.course\_info.course.title}: ({self.title})'

class CourseSkill(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

name = models.CharField(max\_length=255)

def \_\_str\_\_(self):

return f'{self.course\_info.course.title}: ({self.name})'

class CourseStars(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

five\_stars\_count = models.IntegerField(default=0)

four\_stars\_count = models.IntegerField(default=0)

three\_stars\_count = models.IntegerField(default=0)

two\_stars\_count = models.IntegerField(default=0)

one\_stars\_count = models.IntegerField(default=0)

def \_\_str\_\_(self):

return self.course\_info.course.title

class CourseComment(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

created\_at = models.DateTimeField(auto\_now\_add=True)

comment = models.TextField()

stars\_count = models.IntegerField()

def \_\_str\_\_(self):

return f'{self.course\_info.course.title}: ({self.user.user.username})'

class Module(models.Model):

"""

Модуль курса

"""

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

title = models.CharField(max\_length=150)

def \_\_str\_\_(self):

return self.title

class Lesson(models.Model):

"""

Урок модуля к курсу

"""

module = models.ForeignKey(Module, on\_delete=models.CASCADE)

title = models.CharField(max\_length=150)

def \_\_str\_\_(self):

return self.title

class Step(models.Model):

"""

Степ к уроку к модулю к курсу

"""

lesson = models.ForeignKey(Lesson, on\_delete=models.CASCADE)

title = models.CharField(max\_length=150)

content = models.TextField()

def \_\_str\_\_(self):

return self.title

class UserToCourse(models.Model):

"""

Пользователь к курсу. Какие курсы проходит пользователь

"""

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

progress\_points = models.IntegerField()

class UserToModule(models.Model):

"""

Пользователь к Модулю. Какой модуль в курсе проходит пользователь

"""

module = models.ForeignKey(Module, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

class UserToLesson(models.Model):

"""

Пользователь к Уроку. Какой урок в курсе проходит пользователь

"""

lesson = models.ForeignKey(Lesson, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

class UserToStep(models.Model):

"""

Пользователь к Степу. Какой степ в курсе проходит пользователь

"""

step = models.ForeignKey(Step, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)val content: String

)

**Модуль взаимодействия с источниками данных для курсов**

<template>

<div id="main-case">

<HeaderLightBlack/>

<div class="blocks">

<div class="title">

<div class="fixed-container">

<div class="title\_\_body">

<h1>Курсы</h1>

</div>

</div>

</div>

<div class="search">

<div class="fixed-container">

<div class="search\_\_body">

<div class="myself-search">

<form action="" method="get">

<input name="q" type="search" placeholder="Поиск...">

<div class="myself-search\_\_send">

<input type="submit" value="">

</div>

</form>

</div>

</div>

</div>

</div>

<div class="sorting">

<div class="fixed-container">

<div class="sorting\_\_body">

<span class="item-sorting">

<p class="item-sorting\_\_title">По популярности</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По рейтингу</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По цене</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По названию</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По дате добавления</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

</div>

</div>

</div>

<div class="course-raw">

<div class="fixed-container">

<div class="course-raw\_\_body">

<div class="course-raw\_\_course course">

<nuxt-link v-for="course in courses" :to="`/courses/${course.id}`" :key="course.id" class="item-course">

<div class="item-course\_\_wrapper">

<div class="item-course\_\_main-info">

<div class="item-course\_\_icon">

<img :src="course.image" alt="Изображение курса course.title">

</div>

<div class="item-course\_\_text">

<p class="item-course\_\_title">{{ course.title }}</p>

<p class="item-course\_\_author">{{ course.author }}</p>

<p class="item-course\_\_description">{{ course.description }}</p>

</div>

</div>

<div class="item-course\_\_info">

<div class="info-item-course">

<span class="info-item-course\_\_tag">

<img src="img/star.svg" alt="Количество отзывов">

<p>{{ course.rating }}</p>

</span>

<span class="info-item-course\_\_tag">

<img src="img/people.svg" alt="Количество людей записавшиеся на курс">

<p>{{ course.members\_amount }}</p>

</span>

<span class="info-item-course\_\_tag">

<img src="img/clock.svg" alt="Количество часов на прохождение">

<p>{{ course.duration\_in\_minutes }}</p>

</span>

<span v-if="courses.has\_certificate" class="info-item-course\_\_tag">

<img src="img/certificate.svg" alt="Сертификат">

<p>сертификат</p>

</span>

</div>

<span v-if="course.price === 0" class="item-course\_\_price">

<p class="green">Бесплатно</p>

</span>

<span v-else class="item-course\_\_price">

<p class="purple">{{ course.price }}</p>

<img src="/img/rub.svg" alt="Символ рубля">

</span>

</div>

</div>

</nuxt-link>

</div>

<div class="course-raw\_\_filter">

<div class="filter">

<div class="item-filter">

<h6>Цена</h6>

<div class="item-filter\_\_content radio">

<input type="radio" id="price\_\_all" name="price" value="1" checked>

<label for="price\_\_all">Любая</label>

<input type="radio" id="price\_\_free" name="price" value="2">

<label for="price\_\_free">Бесплатно</label>

<input type="radio" id="price\_\_payable" name="price" value="3">

<label for="price\_\_payable">Платно</label>

</div>

</div>

<div class="item-filter" id="status">

<h6>Статус</h6>

<div class="item-filter\_\_content checkbox">

<input type="checkbox" id="checkbox\_\_all" name="status" value="1">

<label for="checkbox\_\_all">Не добавленное</label>

<input type="checkbox" id="checkbox\_\_free" name="status" value="2">

<label for="checkbox\_\_free">Добавленное</label>

<input type="checkbox" id="checkbox\_\_payable" name="status" value="3">

<label for="checkbox\_\_payable">Пройденное</label>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<Footer/>

</div>

</template>

<script>

import axios from "axios";

import HeaderLightBlack from "@/components/HeaderLightBlack"

import Footer from "@/components/Footer";

export default {

name: 'test',

components: {HeaderLightBlack, Footer},

async asyncData(ctx) {

const api = `http://localhost:8000/api/courses`

const {data} = await axios.get(api)

return {

courses: data.results,

}

}

}

</script>

<style lang="scss" scoped>

@import "theme/\_style/course";

</style>

**Модуль отображения списка курсов**

<template>

<div id="main-case">

<HeaderLightBlack/>

<div class="blocks">

<div class="title">

<div class="fixed-container">

<div class="title\_\_body">

<h1>Курсы</h1>

</div>

</div>

</div>

<div class="search">

<div class="fixed-container">

<div class="search\_\_body">

<div class="myself-search">

<form action="" method="get">

<input name="q" type="search" placeholder="Поиск...">

<div class="myself-search\_\_send">

<input type="submit" value="">

</div>

</form>

</div>

</div>

</div>

</div>

<div class="sorting">

<div class="fixed-container">

<div class="sorting\_\_body">

<span class="item-sorting">

<p class="item-sorting\_\_title">По популярности</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По рейтингу</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По цене</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По названию</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По дате добавления</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

</div>

</div>

</div>

<div class="course-raw">

<div class="fixed-container">

<div class="course-raw\_\_body">

<div class="course-raw\_\_course course">

<nuxt-link v-for="course in courses" :to="`/courses/${course.id}`" :key="course.id" class="item-course">

<div class="item-course\_\_wrapper">

<div class="item-course\_\_main-info">

<div class="item-course\_\_icon">

<img :src="course.image" alt="Изображение курса course.title">

</div>

<div class="item-course\_\_text">

<p class="item-course\_\_title">{{ course.title }}</p>

<p class="item-course\_\_author">{{ course.author }}</p>

<p class="item-course\_\_description">{{ course.description }}</p>

</div>

</div>

<div class="item-course\_\_info">

<div class="info-item-course">

<span class="info-item-course\_\_tag">

<img src="img/star.svg" alt="Количество отзывов">

<p>{{ course.rating }}</p>

</span>

<span class="info-item-course\_\_tag">

<img src="img/people.svg" alt="Количество людей записавшиеся на курс">

<p>{{ course.members\_amount }}</p>

</span>

<span class="info-item-course\_\_tag">

<img src="img/clock.svg" alt="Количество часов на прохождение">

<p>{{ course.duration\_in\_minutes }}</p>

</span>

<span v-if="courses.has\_certificate" class="info-item-course\_\_tag">

<img src="img/certificate.svg" alt="Сертификат">

<p>сертификат</p>

</span>

</div>

<span v-if="course.price === 0" class="item-course\_\_price">

<p class="green">Бесплатно</p>

</span>

<span v-else class="item-course\_\_price">

<p class="purple">{{ course.price }}</p>

<img src="/img/rub.svg" alt="Символ рубля">

</span>

</div>

</div>

</nuxt-link>

</div>

<div class="course-raw\_\_filter">

<div class="filter">

<div class="item-filter">

<h6>Цена</h6>

<div class="item-filter\_\_content radio">

<input type="radio" id="price\_\_all" name="price" value="1" checked>

<label for="price\_\_all">Любая</label>

<input type="radio" id="price\_\_free" name="price" value="2">

<label for="price\_\_free">Бесплатно</label>

<input type="radio" id="price\_\_payable" name="price" value="3">

<label for="price\_\_payable">Платно</label>

</div>

</div>

<div class="item-filter" id="status">

<h6>Статус</h6>

<div class="item-filter\_\_content checkbox">

<input type="checkbox" id="checkbox\_\_all" name="status" value="1">

<label for="checkbox\_\_all">Не добавленное</label>

<input type="checkbox" id="checkbox\_\_free" name="status" value="2">

<label for="checkbox\_\_free">Добавленное</label>

<input type="checkbox" id="checkbox\_\_payable" name="status" value="3">

<label for="checkbox\_\_payable">Пройденное</label>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<Footer/>

</div>

</template>

<script>

import axios from "axios";

import HeaderLightBlack from "@/components/HeaderLightBlack"

import Footer from "@/components/Footer";

export default {

name: 'test',

components: {HeaderLightBlack, Footer},

async asyncData(ctx) {

const api = `http://localhost:8000/api/courses`

const {data} = await axios.get(api)

return {

courses: data.results,

}

}

}

</script>

<style lang="scss" scoped>

@import "theme/\_style/course";

</style>

## Разработка пользовательского интерфейса программного средства

При запуске сайта открывается приветственное экран (рис. 8):

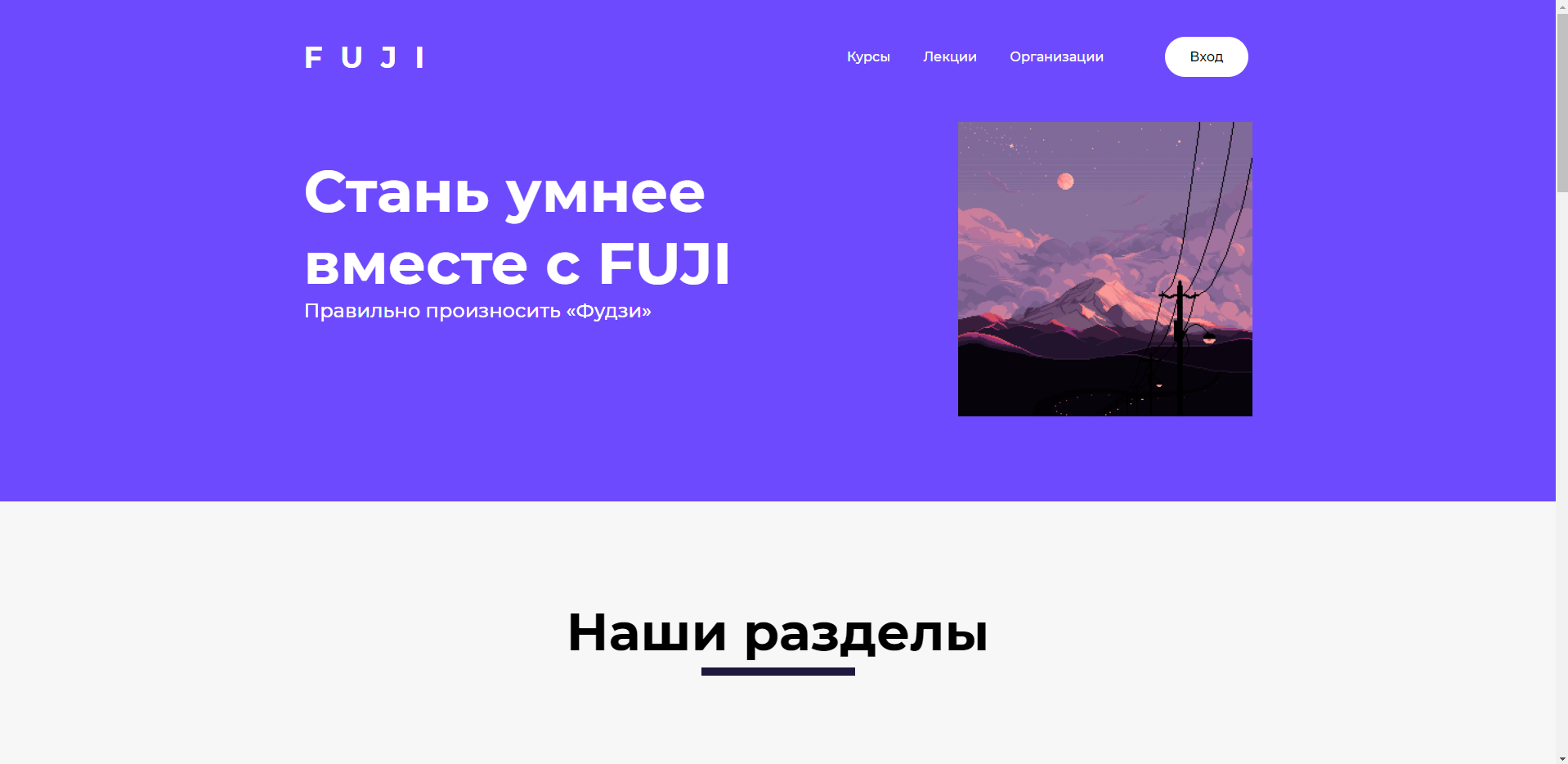


Рисунок 8. Приветственный экран

Отсюда, при нажатии на кнопку «Курсы», вам откроется экран с курсами (рис. 9)

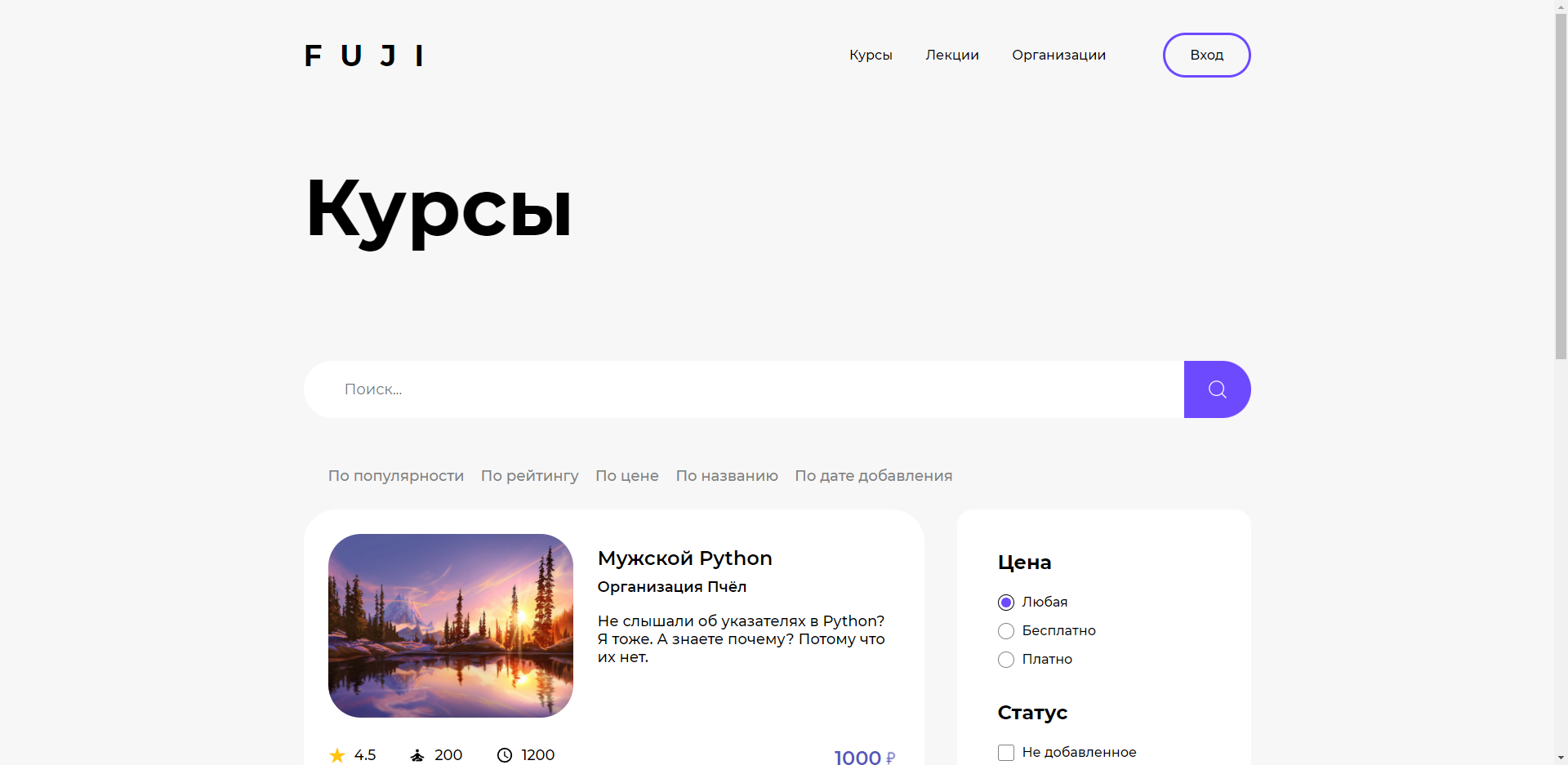


Рисунок 9. Окно с курсами

Выбор пользователем курса, открытие экрана c информацией о курсе (рис. 10, 11).

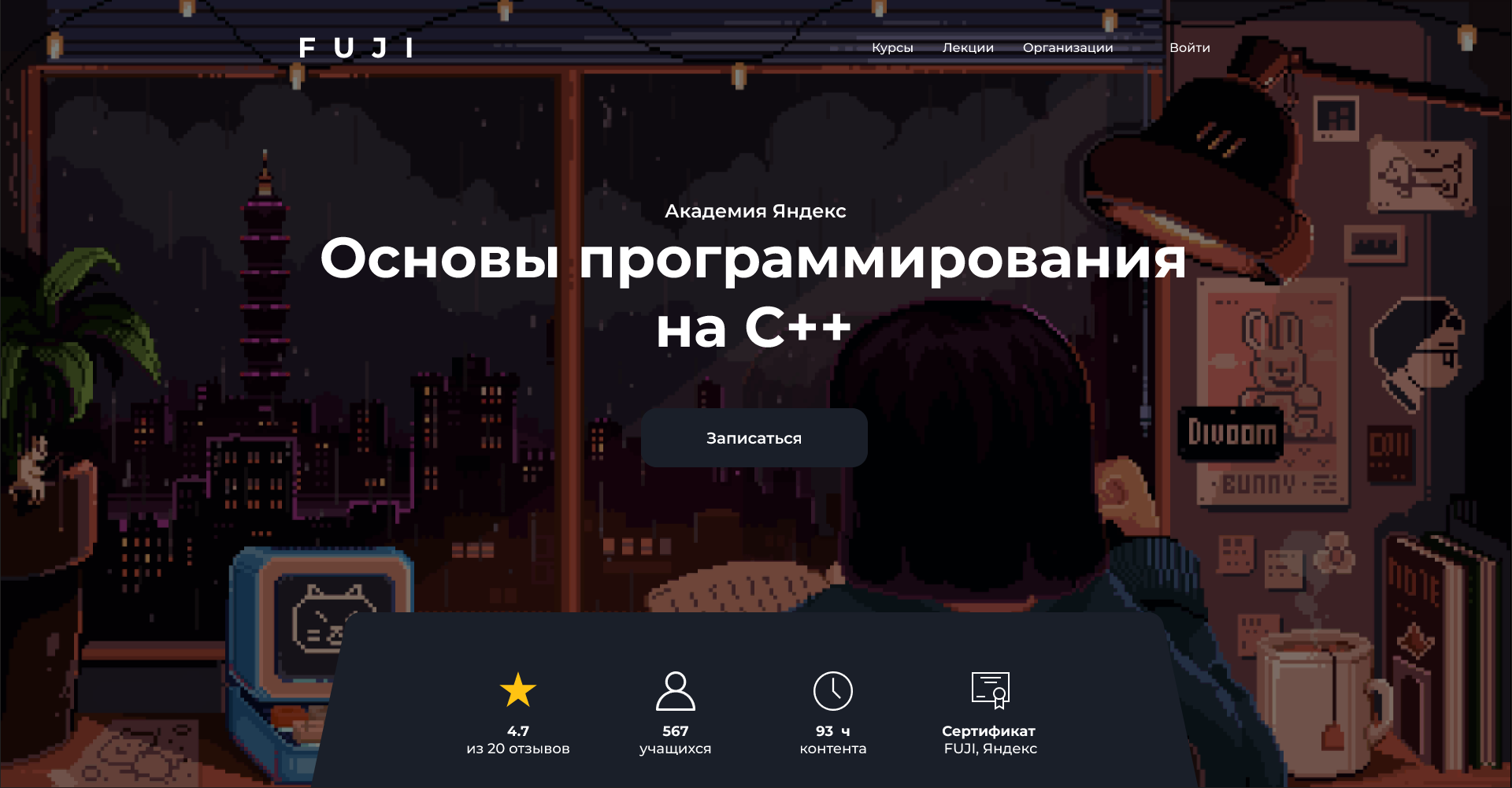


Рисунок 10. Окно с информацией о курсе

## Отладка и тестирование программного средства

Отладка и тестирование программного средства велась в WebStorm. В данной IDE можно воспользоваться консолью для вывода значений переменных в нужных точках программы (рис. 12).

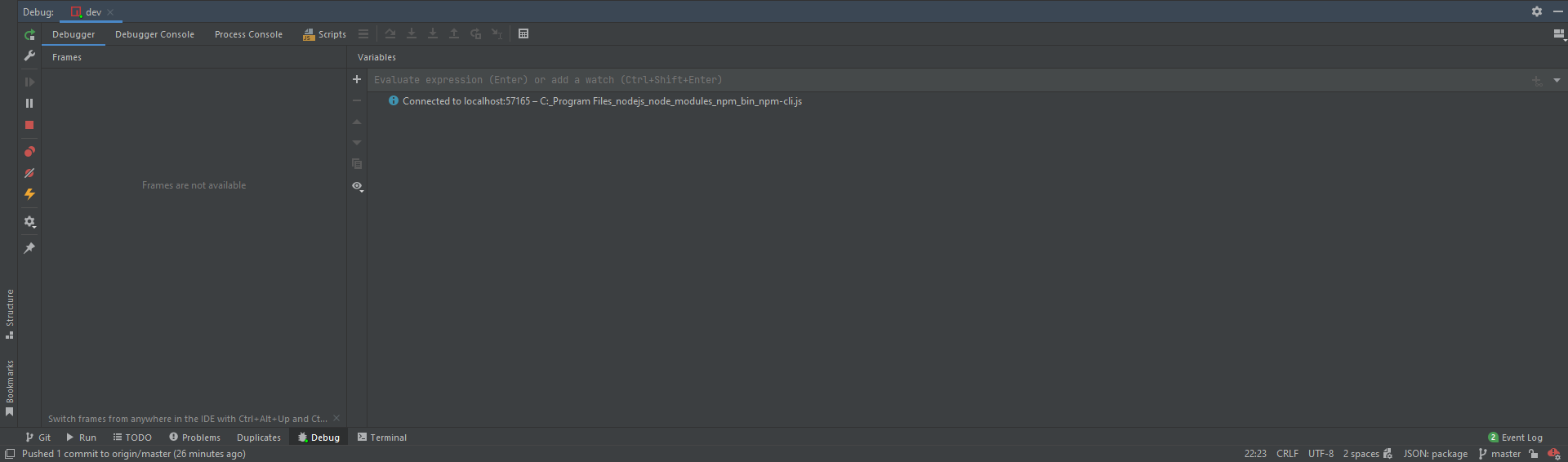


Рисунок 12. Консоль для отладки программы

Также присутствует система автоматизированного тестирования (рис 13).

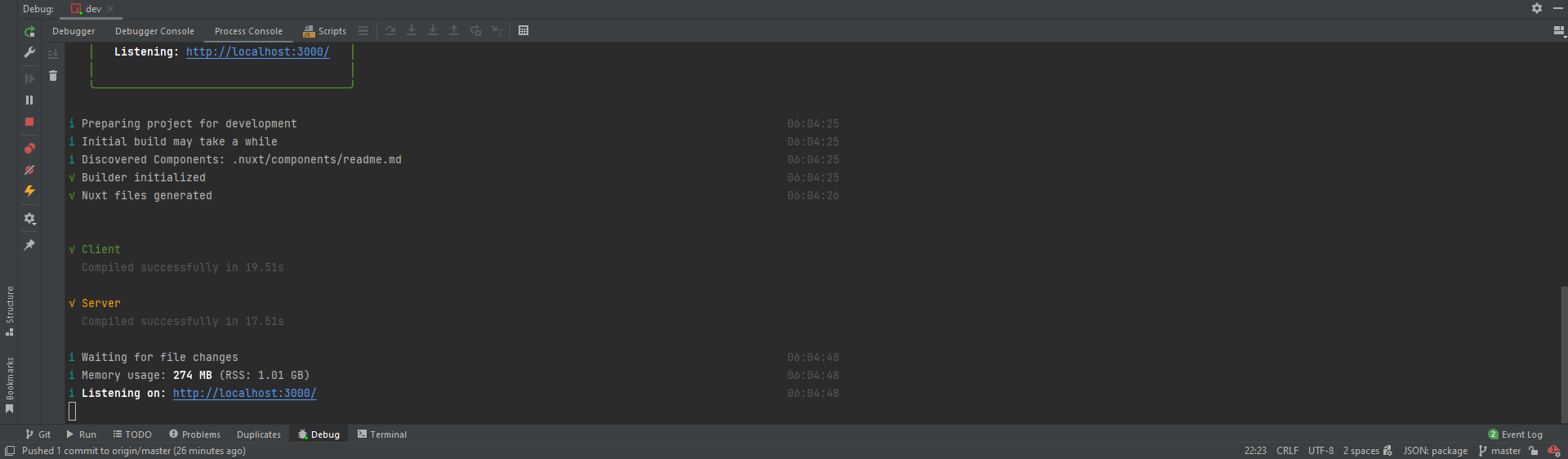


Рисунок 13. Окно автоматизированного тестирования

# Заключение

В ходе производственной практики были получены практические навыки по разработке программных модулей, динамических сайтов и разработки API.

# Список литературы и интернет-ресурсов

1. PostgreSQL

<https://www.postgresql.org/>

1. Django

<https://www.djangoproject.com/>

1. JavaScript

<https://www.javascript.com/>

1. NuxtJs

<https://nuxtjs.org/>

# Дневник практики

Таблица 1. Дневник практики.

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Содержание работы | Отметка о выполнении  работы | Подпись руководителя практики |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Приложение 1. Техническое задание

1. Введение
   1. Наименование программы

Наименование программного продукта – «Fuji» (сайт).

1. Краткая характеристика области

Сайт «Fuji» предназначена для просмотра курсов, составленных пользователями.

1. Основания для разработки

Основанием для разработки является задание на практику.

Наименование темы разработки – «Разработка образовательного портала –Fuji».

1. Назначение разработки

Мобильное приложение «Fuji» будет использоваться для публикации и просмотра курса с возможностью их прохождения.

* 1. Функциональное назначение

Данный сайт позволяет просмотреть текущие курсы, которые находятся на образовательном портале и ознакомиться с их содержанием.

* 1. Эксплуатационное назначение

Сайт «Fuji» планируется использовать в свободном доступе любым пользователем.

1. Требования к программе или программному изделию
   1. Требования к функциональным характеристикам
      1. Требования к составу выполняемых функций

После запуска сайта пользователю отображается главный экран, который является ознакомительной страницей. Далее можно перейти к странице с курсами, при нажатии на каждый из них открывается его содержание.

* + 1. Требование к организации входных и выходных данных

Курсы хранятся на сервере. Студент выбирает курс из предлагаемых списков. После того, как пользователь нашёл нужный курс, нажав по нему, выводятся модули из которых состоит курс. Модули состоят из уроков, которые пользователь может проходить.

* + 1. Требования к временным характеристикам

Требования к временным характеристикам отсутствуют

* + 1. Требования к надёжности

Вероятность безотказной работы системы должна составлять не менее 99.99% при условии исправности версии программы.

* + 1. Требование к обеспечению надёжного функционирования программы

База данных не нуждается в резервном замещении.

Надежное функционирование программы должно быть обеспечено выполнением заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

* организацией бесперебойного питания технических средств;
* Защита информации. Испытания программных средств на наличие компьютерных вирусов.
  + 1. Время восстановления после отказа

Время восстановления после отказа, вызванного неисправностью технических средств, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

* 1. Отказы из-за некорректных действий оператора

Отказы программы вследствие некорректных действий оператора невозможны.

1. Условия эксплуатации

Программа запускается на любом системе позволяющей открыть браузер. Окно программы должно быть открыто на весь экран, должна быть возможность свернуть приложение или запустить любое стороннее программное обеспечение.

* 1. Климатические условия эксплуатации

Специальные условия не требуются.

1. Требования к численности и квалификации

Программа не требует проведения каких-либо видов обслуживания.

* 1. Требования к численности и квалификации персонала

Специальные условия не требуются.

1. Требования к составу и параметрам технических средств
   1. Состав технических средств:
   * IBM PC совместимый с процессором Pentium IV и выше, для комфортной работы тактовая частота д. б. не ниже 2 гГц;
   * ОЗУ более 512 Мбайт, для комфортной работы 1 гБ;
   * 64 МБ видеопамяти и выше, для комфортной работы 256 мБ;
   * Требования к информационной и программной совместимости
   * Наличие любого браузера
   * Подключение к интернету

База данных должна составляется по примеру, который представлен в папке с программой.

* 1. Требования к маркировке и упаковке

Требований нет

* 1. Требования к транспортированию и хранению

Для транспортировки программы нужно интернет соединение.

* 1. Специальные требования

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса, разработанного согласно рекомендациям компании-производителя операционной системы.

* 1. Требования к программной документации

Предварительный состав программной документации:

* техническое задание (включает описание применения);
* программа и методика испытаний;
* руководство системного программиста;
* руководство оператора;

1. Технико-экономические показатели

Программа «Fuji» пригодна для преподавателей и студентов. Так как программа обеспечивает дистанционное просмотр курсов, то спрос на эту программу достаточный, т.к. востребованность дистанционного обучения растёт.

1. Стадии и этапы разработки

Всего этапов разработки:

* Разработка требований
* Проектирование
* Реализация
* Тестирование
* Внедрение

На стадии «Разработка требований» должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии «Проектирование» утверждается интерфейс программы, а также внутренние устройство программы.

На стадии «Реализации» должны быть выполнены перечисленные ниже этапы работ:

* разработка программы;
* разработка программной документации;

На стадии «Тестирование» проверяется работа функций программы и в целом корректность её работы.

На стадии «Внедрение» должен быть выполнено внедрение заказчику на компьютер.

Содержание работ по этапам:

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

* постановка задачи;
* определение и уточнение требований к техническим средствам;
* определение требований к программе;
* определение стадий, этапов и сроков разработки программы и документации на нее;
* согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

* разработка, согласование и утверждение порядка и методики испытаний;
* проведение приемо-сдаточных испытаний;
* корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах заказчика.

1. Порядок контроля и приемки

Приемосдаточные испытания программы должны проводиться согласно разработанной исполнителем и согласованной заказчиком «Программы и методики испытаний».

Ход проведения приемо-сдаточных испытаний заказчик и исполнитель документируют в протоколе испытаний. На основании протокола испытаний исполнитель совместно с заказчиком подписывают акт приемки-сдачи программы в эксплуатацию.

# Приложение 2. Руководство оператора

1. **Назначение программы**
   1. **Функциональное назначение программы**

Данный сайт позволяет просмотреть текущие курсы, которые находятся на образовательном портале и ознакомиться с их содержанием.

Сайт “Fuji” реализует следующие функции:

* Вывод всех курсов
* Вывод всех модулей курса
  1. **Эксплуатационное назначение программы**

Сайт “Fuji” планируется использовать в общем доступе любым человеком.

* + 1. **Функция вывода курсов**

Функция вывода курсов работает в начале программы. Выводит все курсы, которые передаются сайту. Для выбора курса нажмите на картинку с курсом, затем перейдёте на экран в набор выбора модулей.

1. **Условия выполнения программы**
   1. **Минимальный состав аппаратных средств**

Минимальный состав используемых технических (аппаратных) средств:

* IBM PC совместимый с процессором Pentium IV и выше, для комфортной работы тактовая частота д. б. не ниже 2 гГц;
* ОЗУ более 512 Мбайт, для комфортной работы 1 гБ;
* 64 МБ видеопамяти и выше, для комфортной работы 256 мБ;
* Наличие любого браузера
* Подключение к интернету
  1. **Требования к пользователю**

Конечный пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом браузера. Персонал должен быть аттестован на II квалификационную группу по электробезопасности.

1. **Выполнение программы**
   1. **Загрузка и запуск сайта**

Загрузка не требуется.

Запуск сайта осуществляется вводом ссылки в браузер

* 1. **Завершение работы программы**

Для завершения работы сайта достаточно закрыть вкладку.

1. **Сообщения оператору**

Сообщения оператору отсутствуют.

Приложение 3. Руководство программиста

1. **Назначение и условия применения программы**
   1. **Функции программы**

Данный сайт позволяет просмотреть текущие курсы, которые находятся на образовательном портале и ознакомиться с их содержанием.

* 1. **Условия выполнения программы**

Минимальный состав используемых технических (аппаратных) средств:

* IBM PC совместимый с процессором Pentium IV и выше, для комфортной работы тактовая частота д. б. не ниже 2 гГц;
* ОЗУ более 512 Мбайт, для комфортной работы 1 гБ;
* 64 МБ видеопамяти и выше, для комфортной работы 256 мБ;
* Наличие любого браузера
* Подключение к интернету

2. **Характеристики программы**

**2.1. Временные характеристики**

Временные характеристики отсутствуют.



**2.2. Режим работы**

У данного сайта режим работы по запросу.

1. **Обращение к программе**

Обращение к программе производится вводом ссылки в браузер.

Входные и выходные данные:

Входные данные: отсутствуют.

Выходные данные: образовательные материалы.

1. **Сообщения**

Сообщения отсутствуют.

# Приложение 4. Листинг кода

1. Модуль urls. API

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from rest\_framework.urlpatterns import format\_suffix\_patterns

from . import views

router = DefaultRouter()

# router.register('courset', CourseViewSet, basename='courset')

urlpatterns = [

path("", include(router.urls)),

path("register/", views.RegisterView.as\_view()),

path("user/", views.UserView.as\_view()),

path("courses/", views.CourseViewSet.as\_view({'get': 'list'})),

path("courses/<int:pk>/", views.CourseViewSet.as\_view({'get': 'retrieve'})),

]

1. Модуль views. API

# from django.shortcuts import render

from django.core.paginator import Paginator, PageNotAnInteger, EmptyPage

from django.shortcuts import get\_object\_or\_404

from django.urls import path

from rest\_framework.response import Response

from rest\_framework import viewsets, permissions, generics, pagination

from .serializers import CourseSerializer, CourseInfoSerializer

from .serializers import RegisterSerializer, UserSerializer, ProfileSerializer

from .models import Course, Profile, CourseInfo

class PageNumberSetPagination(pagination.PageNumberPagination):

page\_size = 10

page\_size\_query\_param = 'page\_size'

ordering = 'created\_at'

class CourseViewSet(viewsets.ModelViewSet):

lookup\_field = 'slug'

queryset = Course.objects.all()

serializer\_class = CourseSerializer

permission\_classes = [permissions.AllowAny]

pagination\_class = PageNumberSetPagination

def get\_serializer\_class(self):

if self.action == 'list':

return CourseSerializer

elif self.action == 'retrieve':

return CourseInfoSerializer

def retrieve(self, request, pk=None, \*args, \*\*kwargs):

course = Course.objects.get(pk=pk)

serializer = CourseSerializer(course)

course\_info = serializer.get\_info(course.pk)

return Response({

"course": serializer.data,

"info": course\_info

})

class RegisterView(generics.GenericAPIView):

permission\_classes = [permissions.AllowAny]

serializer\_class = RegisterSerializer

def post(self, request, \*args, \*\*kwargs):

serializer = self.get\_serializer(data=request.data)

serializer.is\_valid(raise\_exception=True)

user = serializer.save()

return Response({

'user': UserSerializer(user, context=self.get\_serializer\_context()).data,

'message': "Пользователь успешно создан"

})

class UserView(generics.GenericAPIView):

permission\_classes = [permissions.IsAuthenticated]

serializer\_class = UserSerializer

def get(self, request, \*args, \*\*kwargs):

return Response({

"user": UserSerializer(request.user, context=self.get\_serializer\_context()).data,

})

1. Модуль serializer. API

from rest\_framework import serializers

from .models import Author, Course, Profile

from .models import CourseInfo, CourseFit, CourseSkill, CourseStars, CourseComment

from django.contrib.auth.models import User

class RegisterSerializer(serializers.ModelSerializer):

re\_password = serializers.CharField(write\_only=True)

class Meta:

model = User

fields = ('id', 'username', 'password', 're\_password', 'first\_name', 'last\_name', 'email')

extra\_kwargs = {

'password': {"write\_only": True}

}

def create(self, validated\_data):

username = validated\_data['username']

password = validated\_data['password']

re\_password = validated\_data['re\_password']

first\_name = validated\_data['first\_name']

last\_name = validated\_data['last\_name']

email = validated\_data['email']

if password != re\_password:

raise serializers.ValidationError({'password': "Пароли не совпадают"})

user = User(

username=username,

first\_name=first\_name,

last\_name=last\_name,

email=email

)

user.set\_password(password)

user.save()

profile = Profile(user=user)

profile.save()

return user

class ProfileSerializer(serializers.ModelSerializer):

class Meta:

model = Profile

fields = '\_\_all\_\_'

class UserSerializer(serializers.ModelSerializer):

profile = serializers.SerializerMethodField()

class Meta:

model = User

fields = '\_\_all\_\_'

def get\_profile(self, user):

return ProfileSerializer(user.profile).data

class CourseSerializer(serializers.ModelSerializer):

author = serializers.SlugRelatedField(slug\_field="name", queryset=Author.objects.all())

class Meta:

model = Course

fields = '\_\_all\_\_'

def get\_info(self, pk):

queryset = CourseInfo.objects.get(course\_id=pk)

return CourseInfoSerializer(queryset).data

class CourseInfoSerializer(serializers.ModelSerializer):

fit = serializers.SerializerMethodField()

skill = serializers.SerializerMethodField()

stars = serializers.SerializerMethodField()

comments = serializers.SerializerMethodField()

class Meta:

model = CourseInfo

fields = ['title\_image', 'goal\_description', 'fit', 'skill', 'stars', 'comments']

def get\_fit(self, course\_info):

queryset = CourseFit.objects.filter(course\_info\_id=course\_info.pk)

return CourseFitSerializer(queryset, many=True).data

def get\_skill(self, course\_info):

queryset = CourseSkill.objects.filter(course\_info\_id=course\_info.pk)

return CourseSkillSerializer(queryset, many=True).data

def get\_stars(self, course\_info):

queryset = CourseStars.objects.get(course\_info\_id=course\_info.pk)

return CourseStarsSerializer(queryset).data

def get\_comments(self, course\_info):

queryset = CourseComment.objects.filter(course\_info\_id=course\_info.pk)

return CourseCommentSerializer(queryset, many=True).data

class CourseFitSerializer(serializers.ModelSerializer):

class Meta:

model = CourseFit

fields = ['title', 'description']

# fields = '\_\_all\_\_'

class CourseSkillSerializer(serializers.ModelSerializer):

class Meta:

model = CourseSkill

fields = ['name']

class CourseStarsSerializer(serializers.ModelSerializer):

class Meta:

model = CourseStars

fields = ['five\_stars\_count', 'four\_stars\_count', 'three\_stars\_count', 'two\_stars\_count', 'one\_stars\_count']

class CourseCommentSerializer(serializers.ModelSerializer):

user = serializers.SerializerMethodField()

class Meta:

model = CourseComment

fields = ['user', 'created\_at', 'comment', 'stars\_count']

def get\_user(self, course\_info):

user = User.objects.get(pk=course\_info.pk)

return UserCommentSerializer(user).data

class UserCommentSerializer(serializers.ModelSerializer):

class Meta:

model = User

fields = ['id', 'username', 'first\_name', 'last\_name']

1. Модуль models. API

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Profile(models.Model):

"""

Пользователь

"""

# Расширение класса User (login, password, first\_name, last\_name)

user = models.OneToOneField(User, on\_delete=models.CASCADE)

avatar\_url = models.ImageField(default="default.jpg")

about\_me\_text = models.TextField(blank=True, default="")

def \_\_str\_\_(self):

return f'{self.user}'

class Author(models.Model):

"""

Автор курса

"""

name = models.CharField(max\_length=50)

def \_\_str\_\_(self):

return self.name

class Course(models.Model):

"""

Курс

"""

title = models.CharField(max\_length=100)

description = models.TextField()

price = models.IntegerField()

author = models.ForeignKey(Author, on\_delete=models.CASCADE)

image = models.ImageField()

duration\_in\_minutes = models.IntegerField()

rating = models.FloatField(default=0)

members\_amount = models.IntegerField(default=0)

has\_certificate = models.BooleanField()

max\_progress\_points = models.IntegerField()

def \_\_str\_\_(self):

return self.title

class CourseInfo(models.Model):

"""

Страница курса

"""

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

title\_image = models.ImageField(default="8.jpg")

goal\_description = models.TextField()

def \_\_str\_\_(self):

return self.course.title

class CourseFit(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

title = models.CharField(max\_length=100)

description = models.TextField()

def \_\_str\_\_(self):

return f'{self.course\_info.course.title}: ({self.title})'

class CourseSkill(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

name = models.CharField(max\_length=255)

def \_\_str\_\_(self):

return f'{self.course\_info.course.title}: ({self.name})'

class CourseStars(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

five\_stars\_count = models.IntegerField(default=0)

four\_stars\_count = models.IntegerField(default=0)

three\_stars\_count = models.IntegerField(default=0)

two\_stars\_count = models.IntegerField(default=0)

one\_stars\_count = models.IntegerField(default=0)

def \_\_str\_\_(self):

return self.course\_info.course.title

class CourseComment(models.Model):

course\_info = models.ForeignKey(CourseInfo, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

created\_at = models.DateTimeField(auto\_now\_add=True)

comment = models.TextField()

stars\_count = models.IntegerField()

def \_\_str\_\_(self):

return f'{self.course\_info.course.title}: ({self.user.user.username})'

class Module(models.Model):

"""

Модуль курса

"""

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

title = models.CharField(max\_length=150)

def \_\_str\_\_(self):

return self.title

class Lesson(models.Model):

"""

Урок модуля к курсу

"""

module = models.ForeignKey(Module, on\_delete=models.CASCADE)

title = models.CharField(max\_length=150)

def \_\_str\_\_(self):

return self.title

class Step(models.Model):

"""

Степ к уроку к модулю к курсу

"""

lesson = models.ForeignKey(Lesson, on\_delete=models.CASCADE)

title = models.CharField(max\_length=150)

content = models.TextField()

def \_\_str\_\_(self):

return self.title

class UserToCourse(models.Model):

"""

Пользователь к курсу. Какие курсы проходит пользователь

"""

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

progress\_points = models.IntegerField()

class UserToModule(models.Model):

"""

Пользователь к Модулю. Какой модуль в курсе проходит пользователь

"""

module = models.ForeignKey(Module, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

class UserToLesson(models.Model):

"""

Пользователь к Уроку. Какой урок в курсе проходит пользователь

"""

lesson = models.ForeignKey(Lesson, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

class UserToStep(models.Model):

"""

Пользователь к Степу. Какой степ в курсе проходит пользователь

"""

step = models.ForeignKey(Step, on\_delete=models.CASCADE)

user = models.ForeignKey(Profile, on\_delete=models.CASCADE)

1. Модуль admin. API

from django.contrib import admin

from .models import Profile, Author, Course, Module, Lesson, Step, \

CourseInfo, CourseFit, CourseSkill, CourseStars, CourseComment

class AuthorAdmin(admin.ModelAdmin):

pass

class CourseAdmin(admin.ModelAdmin):

pass

class ProfileAdmin(admin.ModelAdmin):

pass

admin.site.register(Author)

admin.site.register(Course)

admin.site.register(Module)

admin.site.register(Lesson)

admin.site.register(Step)

admin.site.register(CourseInfo)

admin.site.register(CourseFit)

admin.site.register(CourseSkill)

admin.site.register(CourseStars)

admin.site.register(CourseComment)

admin.site.register(Profile)

1. Модуль Footer. components. NuxtJS

<template>

<footer>

<div class="fixed-container">

<div class="footer\_\_body">

<div class="footer\_\_case-logo">

<div class="footer\_\_logo">

<nuxt-link to="/" class="purple logo">FUJI</nuxt-link>

</div>

</div>

<div class="footer\_\_menu">

<p class="footer\_\_title">Меню</p>

<div class="footer\_\_link">

<nuxt-link to="/course">Курсы</nuxt-link>

<nuxt-link to="#">Лекции</nuxt-link>

<nuxt-link to="#">Организации</nuxt-link>

</div>

</div>

<div class="footer\_\_about">

<p class="footer\_\_title">О нас</p>

<div class="footer\_\_link">

<nuxt-link to="/">Контакты</nuxt-link>

<nuxt-link to="/">Вакансии</nuxt-link>

</div>

</div>

<div class="footer\_\_media">

<nuxt-link to="#" class="footer\_\_media-link">

<img src="img/instagram.png" alt="Мы в инсте">

</nuxt-link>

<nuxt-link to="#" class="footer\_\_media-link">

<img src="img/vk.png" alt="Группа в вк">

</nuxt-link>

<nuxt-link to="#" class="footer\_\_media-link">

<img src="img/telegram.svg" alt="Группа в вк">

</nuxt-link>

</div>

</div>

</div>

</footer>

</template>

<script>

export default {

name: "Footer",

}

</script>

<style lang="scss">

@import "../theme/\_style/footer";

</style>

1. Модуль HeaderPurpleWhite. components. NuxtJS

<template>

<header>

<div class="fixed-container">

<div class="header\_\_body">

<div class="case-logo">

<nuxt-link to="/" class="logo">FUJI</nuxt-link>

</div>

<nav class="nav">

<div class="nav\_\_menu">

<nuxt-link to="/course" class="item-nav\_\_menu">Курсы</nuxt-link>

<nuxt-link to="/" class="item-nav\_\_menu">Лекции</nuxt-link>

<nuxt-link to="/" class="item-nav\_\_menu">Организации</nuxt-link>

</div>

<div id="nav\_\_sign">

<nuxt-link to="/signin">Вход</nuxt-link>

</div>

</nav>

</div>

</div>

</header>

</template>

<script>

export default{

name: "HeaderPurpleWhite",

}

</script>

<style lang="scss" scoped>

@import "../theme/\_style/header";

@include header-light($fc-white, $fc-black, $bgc-purple);

</style>

1. Модуль HeaderLightBlack. components. NuxtJS

<template>

<header>

<div class="fixed-container">

<div class="header\_\_body">

<div class="case-logo">

<nuxt-link to="/" class="logo">FUJI</nuxt-link>

</div>

<nav class="nav">

<div class="nav\_\_menu">

<nuxt-link to="/course" class="item-nav\_\_menu">Курсы</nuxt-link>

<nuxt-link to="/" class="item-nav\_\_menu">Лекции</nuxt-link>

<nuxt-link to="/" class="item-nav\_\_menu">Организации</nuxt-link>

</div>

<div id="nav\_\_sign">

<nuxt-link to="/signin">Вход</nuxt-link>

</div>

</nav>

</div>

</div>

</header>

</template>

<script>

export default{

name: "HeaderLightBlack",

}

</script>

<style lang="scss" scoped>

@import "../theme/\_style/header";

@include header-black($fc-black, $bgc-purple, $bgc-light);

</style>

1. Модуль HeaderLightBlack. components. NuxtJS

<template>

<header>

<div class="fixed-container">

<div class="header\_\_body">

<div class="case-logo">

<nuxt-link to="/" class="logo">FUJI</nuxt-link>

</div>

<nav class="nav">

<div class="nav\_\_menu">

<nuxt-link to="/course" class="item-nav\_\_menu">Курсы</nuxt-link>

<nuxt-link to="/" class="item-nav\_\_menu">Лекции</nuxt-link>

<nuxt-link to="/" class="item-nav\_\_menu">Организации</nuxt-link>

</div>

<div id="nav\_\_sign">

<nuxt-link to="/signin">Вход</nuxt-link>

</div>

</nav>

</div>

</div>

</header>

</template>

<script>

export default{

name: "HeaderLightBlack",

}

</script>

<style lang="scss" scoped>

@import "../theme/\_style/header";

@include header-rofl($fc-black, $fc-white, $bgc-empty);

</style>

1. Модуль index. pages. NuxtJS

<template>

<div id="main-case">

<HeaderPurpleWhite />

<div class="blocks">

<div class="base">

<div class="fixed-container">

<div class="base\_\_body">

<div class="base\_\_info">

<h2 class="base\_\_title">Стань умнее вместе с FUJI</h2>

<p class="base\_\_description">Правильно произносить «Фудзи»</p>

</div>

<div class="base\_\_icon">

<img src="img/mountain.gif" alt="Горы">

</div>

</div>

</div>

</div>

<div class="section">

<div class="fixed-container">

<div class="section\_\_body">

<div class="section\_\_header">

<h3 class="section\_\_title">Наши разделы</h3>

<div class="line"/>

</div>

<div class="section\_\_content">

<div class="section\_\_card">

<div class="item-section">

<div class="item-section\_\_text">

<h5 class="item-section\_\_title">Курсы</h5>

<p class="item-section\_\_description">Серия из лекций позволяющая изучить какой-либо предмет или обучиться навыку</p>

<div class="item-section\_\_link">

<nuxt-link to="/course">Перейти</nuxt-link>

</div>

</div>

<div class="item-section\_\_icon">

<img src="img/book.svg" alt="Карандаш">

</div>

</div>

</div>

<div class="section\_\_card">

<div class="item-section">

<div class="item-section\_\_text">

<h5 class="item-section\_\_title">Организации</h5>

<p class="item-section\_\_description">Это люди, компании, школы, которые создают для вас весь контент (курсы, лекции)</p>

<div class="item-section\_\_link">

<nuxt-link to="#">Перейти</nuxt-link>

</div>

</div>

<div class="item-section\_\_icon">

<img src="img/pencil.svg" alt="Карандаш">

</div>

</div>

</div>

<div class="section\_\_card">

<div class="item-section">

<div class="item-section\_\_text">

<h5 class="item-section\_\_title">Лекции</h5>

<p class="item-section\_\_description">Структурированный разбор темы</p>

<div class="item-section\_\_link">

<nuxt-link to="#">Перейти</nuxt-link>

</div>

</div>

<div class="item-section\_\_icon">

<img src="img/sheet.svg" alt="Карандаш">

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<div class="mobile">

<div class="fixed-container">

<div class="mobile\_\_body">

<div class="mobile\_\_text">

<h4 class="mobile\_\_title">Изучайте курсы везде</h4>

<p class="mobile\_\_description">FUJI доступен и на Android</p>

<div class="mobile\_\_google-play">

<img src="img/google\_play.svg" alt="Доступно в Google Play">

</div>

</div>

<div class="mobile\_\_icon">

<img src="img/android.svg" alt="android">

</div>

</div>

</div>

</div>

</div>

<Footer />

</div>

</template>

<script>

import HeaderPurpleWhite from "@/components/HeaderPurpleWhite"

import Footer from "@/components/Footer";

export default {

name: 'IndexPage',

components: {HeaderPurpleWhite, Footer}

}

</script>

<style lang="scss" scoped>

@import "theme/\_style/index";

</style>

1. Модуль course. pages. NuxtJS

<template>

<div id="main-case">

<HeaderLightBlack/>

<div class="blocks">

<div class="title">

<div class="fixed-container">

<div class="title\_\_body">

<h1>Курсы</h1>

</div>

</div>

</div>

<div class="search">

<div class="fixed-container">

<div class="search\_\_body">

<div class="myself-search">

<form action="" method="get">

<input name="q" type="search" placeholder="Поиск...">

<div class="myself-search\_\_send">

<input type="submit" value="">

</div>

</form>

</div>

</div>

</div>

</div>

<div class="sorting">

<div class="fixed-container">

<div class="sorting\_\_body">

<span class="item-sorting">

<p class="item-sorting\_\_title">По популярности</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По рейтингу</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По цене</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По названию</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

<span class="item-sorting">

<p class="item-sorting\_\_title">По дате добавления</p>

<img src="img/big\_small.svg" alt="" class="item-sorting\_\_icon">

</span>

</div>

</div>

</div>

<div class="course-raw">

<div class="fixed-container">

<div class="course-raw\_\_body">

<div class="course-raw\_\_course course">

<nuxt-link v-for="course in courses" :to="`/courses/${course.id}`" :key="course.id" class="item-course">

<div class="item-course\_\_wrapper">

<div class="item-course\_\_main-info">

<div class="item-course\_\_icon">

<img :src="course.image" alt="Изображение курса course.title">

</div>

<div class="item-course\_\_text">

<p class="item-course\_\_title">{{ course.title }}</p>

<p class="item-course\_\_author">{{ course.author }}</p>

<p class="item-course\_\_description">{{ course.description }}</p>

</div>

</div>

<div class="item-course\_\_info">

<div class="info-item-course">

<span class="info-item-course\_\_tag">

<img src="img/star.svg" alt="Количество отзывов">

<p>{{ course.rating }}</p>

</span>

<span class="info-item-course\_\_tag">

<img src="img/people.svg" alt="Количество людей записавшиеся на курс">

<p>{{ course.members\_amount }}</p>

</span>

<span class="info-item-course\_\_tag">

<img src="img/clock.svg" alt="Количество часов на прохождение">

<p>{{ course.duration\_in\_minutes }}</p>

</span>

<span v-if="courses.has\_certificate" class="info-item-course\_\_tag">

<img src="img/certificate.svg" alt="Сертификат">

<p>сертификат</p>

</span>

</div>

<span v-if="course.price === 0" class="item-course\_\_price">

<p class="green">Бесплатно</p>

</span>

<span v-else class="item-course\_\_price">

<p class="purple">{{ course.price }}</p>

<img src="/img/rub.svg" alt="Символ рубля">

</span>

</div>

</div>

</nuxt-link>

</div>

<div class="course-raw\_\_filter">

<div class="filter">

<div class="item-filter">

<h6>Цена</h6>

<div class="item-filter\_\_content radio">

<input type="radio" id="price\_\_all" name="price" value="1" checked>

<label for="price\_\_all">Любая</label>

<input type="radio" id="price\_\_free" name="price" value="2">

<label for="price\_\_free">Бесплатно</label>

<input type="radio" id="price\_\_payable" name="price" value="3">

<label for="price\_\_payable">Платно</label>

</div>

</div>

<div class="item-filter" id="status">

<h6>Статус</h6>

<div class="item-filter\_\_content checkbox">

<input type="checkbox" id="checkbox\_\_all" name="status" value="1">

<label for="checkbox\_\_all">Не добавленное</label>

<input type="checkbox" id="checkbox\_\_free" name="status" value="2">

<label for="checkbox\_\_free">Добавленное</label>

<input type="checkbox" id="checkbox\_\_payable" name="status" value="3">

<label for="checkbox\_\_payable">Пройденное</label>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<Footer/>

</div>

</template>

<script>

import axios from "axios";

import HeaderLightBlack from "@/components/HeaderLightBlack"

import Footer from "@/components/Footer";

export default {

name: 'test',

components: {HeaderLightBlack, Footer},

async asyncData(ctx) {

const api = `http://localhost:8000/api/courses`

const {data} = await axios.get(api)

return {

courses: data.results,

}

}

}

</script>

<style lang="scss" scoped>

@import "theme/\_style/course";

</style>

1. Модуль \_fonts. theme. NuxtJS

@import "theme/\_mixins/font-face";

// font-family: 'Montserrat-Regular', sans-serif

@include font('Montserrat-Regular', '/fonts/Montserrat/Montserrat-Regular.ttf');

// font-family: 'Montserrat-Medium', sans-serif

@include font('Montserrat-Medium', '/fonts/Montserrat/Montserrat-Medium.ttf');

// font-family: 'Montserrat-SemiBold', sans-serif

@include font('Montserrat-SemiBold', '/fonts/Montserrat/Montserrat-SemiBold.ttf');

// font-family: 'Montserrat-Bold', sans-serif

@include font('Montserrat-Bold', 'static/fonts/Montserrat/Montserrat-Bold.ttf');

1. Модуль \_font-face. theme. NuxtJS

@mixin font($f-name, $f-url) {

@font-face {

font-family: $f-name;

src: url($f-url);

font-display: swap;

}

}

1. Модуль \_\_header-style-face. theme. NuxtJS

// $f-color-main: color text

// $f-color: button color text

// $bg-color: background color

@mixin header-light($f-color-main, $f-color-button, $bg-color) {

@include header-style-default($f-color-main, $f-color-main, $bg-color);

header #nav\_\_sign a {

padding: 15px 30px;

background-color: $f-color-main;

border-radius: 40px;

border: $bg-color 5px solid;

transition-duration: .3s;

color: $f-color-button;

&:hover {

border: $f-color-main 5px solid;

transition-duration: .3s;

}

}

}

@mixin header-black($f-color-main, $f-color-button, $bg-color){

@include header-style-default($f-color-main, $f-color-button, $bg-color);

header #nav\_\_sign a{

padding: 15px 30px;

border-radius: 40px;

border: $f-color-button 3px solid;

transition-duration: .3s;

&:hover {

background-color: $f-color-button;

color: $bg-color;

transition-duration: .3s;

}

}

}

@mixin header-rofl($f-color-main, $f-color-button, $bg-color){

@include header-style-default($f-color-main, $f-color-button, $bg-color);

.fixed-container {

position: relative;

.header\_\_body {

width: 100%;

position: absolute;

}

}

header #nav\_\_sign a{

padding: 15px 30px;

border-radius: 40px;

transition-duration: .3s;

&:hover {

background-color: $f-color-main;

color: $f-color-button;

transition-duration: .3s;

}

}

}

@mixin header-style-default($f-color-main, $f-color-button, $bg-color) {

header {

background-color: $bg-color;

a{ color: $f-color-main;}

.nav {

&\_\_menu a {

border-bottom: $bg-color 2px solid;

transition-duration: .3s;

&:hover {

transition-duration: .3s;

border-bottom: $f-color-button 2px solid;

padding-bottom: 3px;

}

}

#nav\_\_sign {display: flex;}

}

}

}

1. Модуль \_media. theme. NuxtJS

$lg-desktop-width: 1200px;

$md-desktop-width: 992px;

$sm-tablets-width: 768px;

$sm-mobile-width: 480px;

$extra-sm-mobile-width: 320px;

/\*========== Desktop First ==========\*/

/\* Large devices | 1200px or less \*/

@mixin lg-desktop {

@media only screen and (max-width: #{$lg-desktop-width}) {

@content

}

}

/\* Medium devices | 992px or less \*/

@mixin md-desktop {

@media only screen and (max-width: #{$md-desktop-width}) {

@content

}

}

/\* Small tablet devices | 768px or less \*/

@mixin sm-tablets {

@media only screen and (max-width: #{$sm-tablets-width}) {

@content

}

}

/\* Small mobile devices | 480px or less \*/

@mixin sm-mobile {

@media only screen and (max-width: #{$sm-mobile-width}) {

@content

}

}

/\* Extra small mobile devices | 320px or less \*/

@mixin esm-mobile {

@media only screen and (max-width: #{$extra-sm-mobile-width}) {

@content

}

}

1. Модуль \_course. theme. NuxtJS

.title {

background-color: $bgc-light;

padding-top: 100px;

}

.search {

background-color: $bgc-light;

padding-top: 130px;

.search\_\_body .myself-search form {

width: 100%;

input[name='q'] {

font-size: 18px;

font-family: $f-small;

width: 93%;

background-color: $bgc-white;

padding: 24px 10px 24px 50px;

border-radius: 40px 0 0 40px;

border: white 0 solid;

outline: none;

}

.myself-search\_\_send {

width: 7%;

height: 70px;

float: right;

border-radius: 0 40px 40px 0;

margin-bottom: 60px;

background-color: $bgc-purple;

input {

width: 100%;

height: 100%;

border: none;

background: url(static/img/search\_\_white.png) no-repeat 50% 50%;

}

}

}

}

.sorting {

background-color: $bgc-light;

padding-top: 60px;

&\_\_body {

margin-left: 30px;

display: flex;

.item-sorting {

display: flex;

cursor: pointer;

color: $fc-disable;

&:not(&:last-child) {

margin-right: 20px;

}

img {

margin-left: 8px;

display: none;

}

&\_activate {

color: $fc-black;

img {

display: block;

}

}

}

}

}

.course-raw {

background-color: $bgc-light;

padding-top: 30px;

.course-raw\_\_body {

display: grid;

grid: 600px auto;

grid-template-columns: 1fr 1fr;

.course-raw\_\_course {

width: 760px;

margin-bottom: 30px;

.item-course\_\_wrapper {

background-color: #fff;

border-radius: 40px;

padding: 30px 30px 25px;

color: #000;

transition-duration: .3s;

margin-bottom: 40px;

&:hover {

transition-duration: .2s;

box-shadow: 0 14px 28px $box-shadow-dark-light, 0 10px 10px $box-shadow-dark;

}

.item-course\_\_main-info {

display: flex;

.item-course\_\_icon {

width: 300px;

height: 225px;

border-radius: 40px;

display: flex;

justify-content: center;

align-items: center;

overflow: hidden;

img {

width: auto;

height: 100%;

background-repeat: no-repeat;

background-size: cover;

background-position: center;

}

}

.item-course\_\_text {

width: 370px;

margin-left: auto;

.item-course\_\_title {

font-family: $f-subheading;

font-size: $fs-lg;

margin-top: 15px;

}

.item-course\_\_author {

font-family: $f-subheading;

margin-top: 10px;

}

.item-course\_\_description {

margin-top: 20px;

}

}

}

.item-course\_\_info {

display: flex;

flex-wrap: wrap;

align-content: flex-start;

margin-top: 35px;

margin-bottom: 25px;

}

.info-item-course {

display: flex;

.info-item-course\_\_tag {

display: flex;

&:not(:last-child) {

margin-right: 40px;

}

img {

width: 22px;

height: 22px;

}

p {

margin-left: 10px;

}

}

}

.item-course\_\_price {

display: flex;

margin-left: auto;

img {

width: 22px;

height: 22px;

margin: auto 0;

}

.green {

color: $fc-active;

}

.purple {

color: $fc-money;

font-family: $f-subheading;

font-size: $fs-lg;

}

}

}

}

.filter {

margin-left: 40px;

background-color: $bgc-white;

border-radius: 20px;

padding: 50px;

position: sticky;

top: 20px;

margin-bottom: 70px;

.item-filter:not(:last-child) {

margin-bottom: 40px;

}

.item-filter\_\_content {

margin-top: 10px;

display: flex;

flex-direction: column;

input {

position: absolute;

z-index: -1;

opacity: 0;

& + label {

display: inline-flex;

user-select: none;

margin-top: 15px;

max-width: 220px;

&::before {

content: '';

display: inline-block;

width: 18px;

height: 18px;

flex-shrink: 0;

flex-grow: 0;

border: 1px solid $fc-disable;

border-radius: 50%;

margin-right: 10px;

background-repeat: no-repeat;

background-position: center center;

}

}

&:not(:disabled) {

/\* При наведение на input \*/

&:not(:checked) + label:hover::before {

border-color: $bgc-black;

}

/\* Для активной input (при нажатии на неё) \*/

&:active + label::before {

border-color: $bgc-purple;

background-color: $bgc-purple;

}

}

&:focus {

& + label::before {

box-shadow: 0 0 0 0.2rem $box-shadow-purple;

}

&:not(:checked) + label::before {

border-color: $bgc-light-purple;

}

}

&:checked + label::before {

border-color: $bgc-black;

background-color: $bgc-white;

}

&:disabled + label::before {

background-color: $bgc-light;

}

}

}

.checkbox input {

/\* input[checkbox] загругление границ \*/

& + label::before {

border-radius: 3px;

}

/\* input[checkbox] внутренний изображение внутри input \*/

&:checked + label::before {

background-image: url(static/img/checkbox\_active.png);

}

}

.radio input {

/\* input[radio] загругление границ \*/

& + label::before {

border-radius: 50%;

background-size: 12px;

}

/\* input[checkbox] внутренний изображение внутри input \*/

&:checked + label::before {

background-image: url(static/img/ellipse\_purple.png);

}

}

}

}

}

1. Модуль \_footer. theme. NuxtJS

footer {

background-color: $bgc-dark-purple;

.footer\_\_body {

display: flex;

padding: 150px 0;

.footer\_\_logo {

display: flex;

a.logo {

color: $fc-white;

font-size: $fs-h5;

padding: 90px 100px;

}

}

.footer\_\_menu {

margin-left: 240px;

}

.footer\_\_about {

margin-left: 70px;

}

.footer\_\_title {

color: $fc-purple-gray;

font-family: $f-subheading;

}

.footer\_\_link {

display: flex;

flex-direction: column;

margin-top: 20px;

width: 150px;

a {

color: $fc-white;

transition-duration: .2s;

&:not(&:first-child) {

margin-top: 20px;

}

&:hover {

transition-duration: .2s;

border-left: $bgc-purple 4px solid;

padding-left: 5px;

}

}

}

.footer\_\_media {

margin-left: 70px;

display: flex;

flex-direction: column;

&-link:not(&-link:first-child) {

margin-top: 10px;

}

}

}

.purple {

background-color: $bgc-purple;

}

.green {

background-color: $bgc-green;

}

.blue {

background-color: $bgc-blue;

}

.red {

background-color: $bgc-red;

}

}

1. Модуль \_header. theme. NuxtJS

header {

padding-top: 40px;

.header\_\_body {

display: flex;

.case-logo {

margin: auto 0;

a.logo {

font-size: $fs-h5;

}

}

.nav {

display: flex;

align-items: center;

margin-left: auto;

a {

font-size: $fs-rg;

}

&\_\_menu a{

&:not(&:first-child) {

margin-left: 36px;

}

}

#nav\_\_sign {

margin-left: 70px;

a {

margin: auto;

}

}

}

}

}

1. Модуль \_index. theme. NuxtJS

.base {

background-color: $bgc-purple;

padding: 50px 0 100px;

&\_\_body {

display: flex;

color: $fc-white;

.base\_\_info {

max-width: 765px;

margin: 40px 0;

p {

font-size: $fs-lg;

}

}

.base\_\_icon {

margin-left: auto;

img {

width: 360px;

height: 360px;

}

}

}

}

.section {

background-color: $bgc-light;

padding: 120px 0 150px;

&\_\_header {

margin: 0 150px;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

h3 {

text-align: center;

}

.line {

margin-top: 5px;

width: 188px;

height: 10px;

background-color: $bgc-dark-purple;

}

}

&\_\_content {

margin-top: 120px;

.section\_\_card {

background-color: $bgc-dark-purple;

border-radius: 20px;

transition-duration: .3s;

&:hover {

transform: scale(1.02);

}

&:not(&:last-child) {

margin-bottom: 40px;

}

.item-section {

margin: 0 160px;

display: flex;

&\_\_text {

width: 440px;

margin: 100px 0 85px;

color: $fc-white;

.item-section\_\_description {

margin-top: 20px;

}

.item-section\_\_link {

display: flex;

margin-top: 40px;

a {

border-radius: 20px;

border: $bgc-white 1px solid;

color: $fc-white;

padding: 20px 60px;

transition-duration: .3s;

&:hover {

padding: 20px 80px;

transition-duration: .3s;

font-family: $f-heading;

border: $bgc-purple 1px solid;

color: $bgc-purple;

}

}

}

}

&\_\_icon {

margin: auto 0 auto auto;

}

}

}

}

}

.mobile {

background-color: $bgc-purple;

&\_\_body {

display: flex;

padding: 180px 0;

.mobile\_\_text {

color: $fc-white;

margin: auto 0;

.mobile\_\_description {

margin-top: 10px;

font-size: $fs-slg;

}

.mobile\_\_google-play {

margin-top: 50px;

img {

width: 270px;

}

}

}

.mobile\_\_icon {

margin-left: auto;

img {

width: 315px;

}

}

}

}

1. Модуль \_vars. theme. NuxtJS

// ========== Font ==========

// Font style

$f-default: 'Montserrat-Medium', sans-serif;

$f-heading: 'Montserrat-Bold', sans-serif;

$f-subheading: 'Montserrat-SemiBold', sans-serif;

$f-small: 'Montserrat-Regular', sans-serif;

// Font size heading

$fs-h1: 96px;

$fs-h2: 72px;

$fs-h3: 64px;

$fs-h4: 48px;

$fs-h5: 36px;

$fs-h6: 24px;

// Font size paragraph

// [Large, SemiLarge, Medium, Regular, Small, ExtraSmall]

$fs-lg: 24px;

$fs-slg: 20px;

$fs-md: 18px;

$fs-rg: 16px;

$fs-sm: 14px;

$fs-esm: 12px;

// Font size link

$fs-a: $fs-md;

// Font color

$fc-white: #FFF;

$fc-black: #000;

$fc-active: #2ED573;

$fc-money: #5758BB;

$fc-disable: #828282;

$fc-purple-gray: #524C6D;

// ========== Background ==========

// Background color

$bgc-white: #FFF;

$bgc-black: #000;

$bgc-light: #F7F7F8;

$bgc-purple: #6E4AFF;

$bgc-dark-purple: #1F173F;

$bgc-light-purple: #9F8BEE;

$bgc-green: #2ED573;

$bgc-red: #ff6b6b;

$bgc-blue: #0abde3;

$bgc-empty: #00000000;

// ========== Width ==========

// Max width

$max-width: 1160px;

// ========== box-shadow ==========

$box-shadow-purple: rgba(159, 139, 238, 0.25);

$box-shadow-dark-light: rgba(0, 0, 0, 0.1);

$box-shadow-dark: rgba(0, 0, 0, 0.15);

1. Модуль default. theme. NuxtJS

@import "\_fonts/fonts";

@import "\_vars/vars";

/\* ----------------------------------------------------------------

/\* >>> TABLE OF CONTENTS:

/\* ----------------------------------------------------------------

/\* 1. Null

/\* 2. Body

/\* 3. Container

/\* 4. Text

/\* --------------------------------------------------------------\*/

/\* ----------------------------------------------------------------

/\* 1. Null

/\* --------------------------------------------------------------\*/

\*, \*::before, \*::after{

margin: 0;

padding: 0;

}

a { text-decoration: none; }

/\* ----------------------------------------------------------------

/\* 2. Body

/\* --------------------------------------------------------------\*/

body {

font-family: $f-default;

}

/\* ----------------------------------------------------------------

/\* 3. Container

/\* --------------------------------------------------------------\*/

.fixed-container {

max-width: $max-width;

margin: 0 auto;

}

.fixed-container-big {

display: flex;

margin: 0 200px;

}

/\* ----------------------------------------------------------------

/\* 4. Text

/\* --------------------------------------------------------------\*/

h1, h2, h3, h4, h5, h6 {font-family: $f-heading}

h1 {font-size: $fs-h1;}

h2 {font-size: $fs-h2;}

h3 {font-size: $fs-h3;}

h4 {font-size: $fs-h4;}

h5 {font-size: $fs-h5;}

h6 {font-size: $fs-h6;}

p {font-size: $fs-md;}

a {font-size: $fs-a;}

a.logo {

letter-spacing: 0.6em;

font-family: $f-heading;

}

\*::selection {

background: $bgc-red;

color: $fc-white;

}