

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им. В.Г.Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных
систем

Расчётно-графическое задание
дисциплина: «Технологии Web-программирования»
тема: «Сайт интернет-магазина одежды»

Выполнил: студент группы ВТ-41
Шкорба В. С.
Проверил: Картамышев С.В.

Белгород 2020

Оглавление

Постановка задачи и выбор средств разработки	3
1. Разработка шаблонов страниц	4
2. Перенос макетов на Vue JS	9
3. Разворачивание базового приложения Yii2	15
4. Проектирование и разработка базы данных.....	18
5. Разработка REST API	22
6. Работа с HTTP запросами	24
Вывод.....	29
Приложение.....	30

Цель работы

Целью данной работы является ознакомление с принципами веб-разработки в общем, изучение языка разметки HTML и языка стилей CSS, ознакомление с основами разработки frontend- и backend-приложений, знакомство с docker, изучение основ взаимодействия backend-приложения с базой данных frontend- и backend-приложений между собой посредством Ajax запросов.

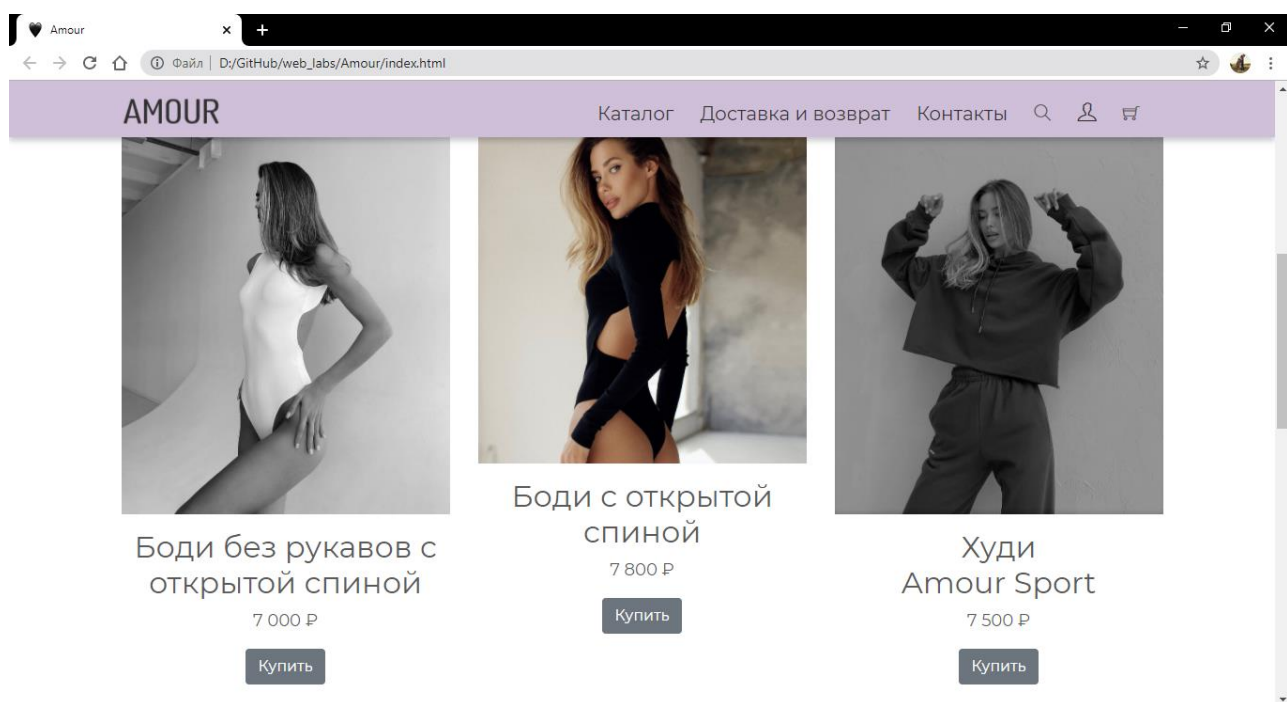
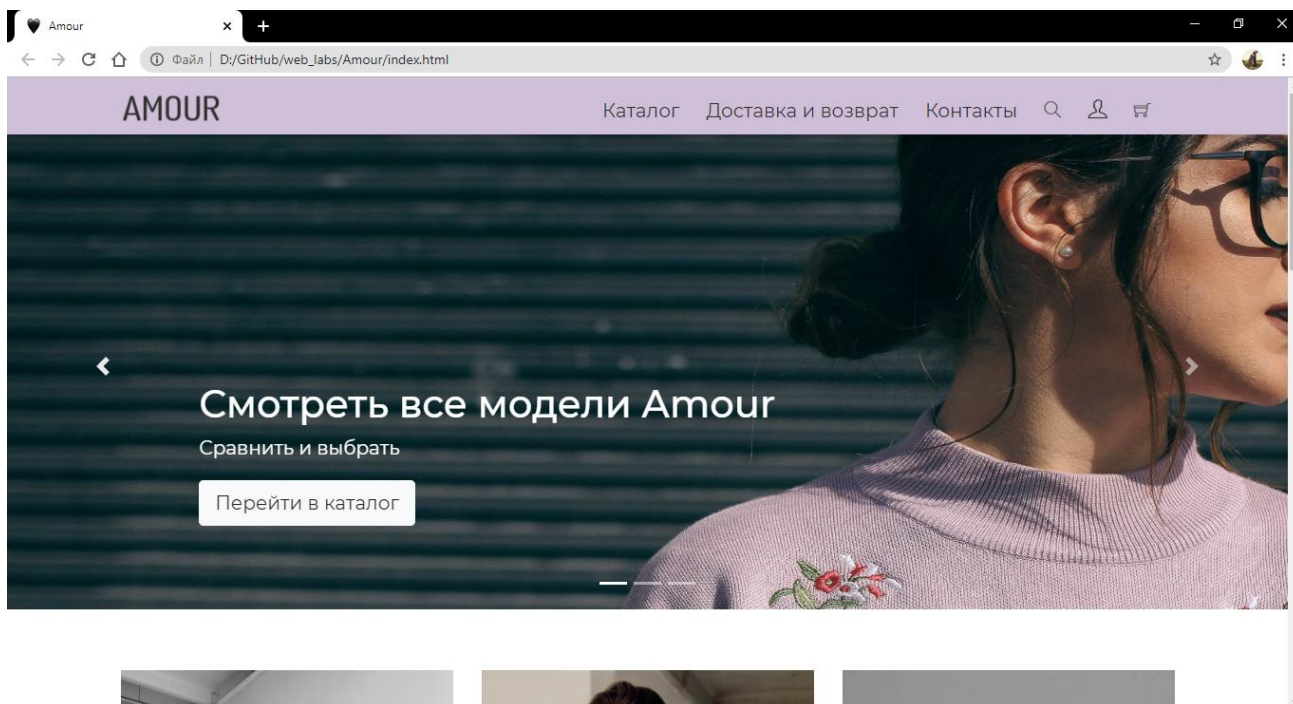
Постановка задачи и выбор средств разработки

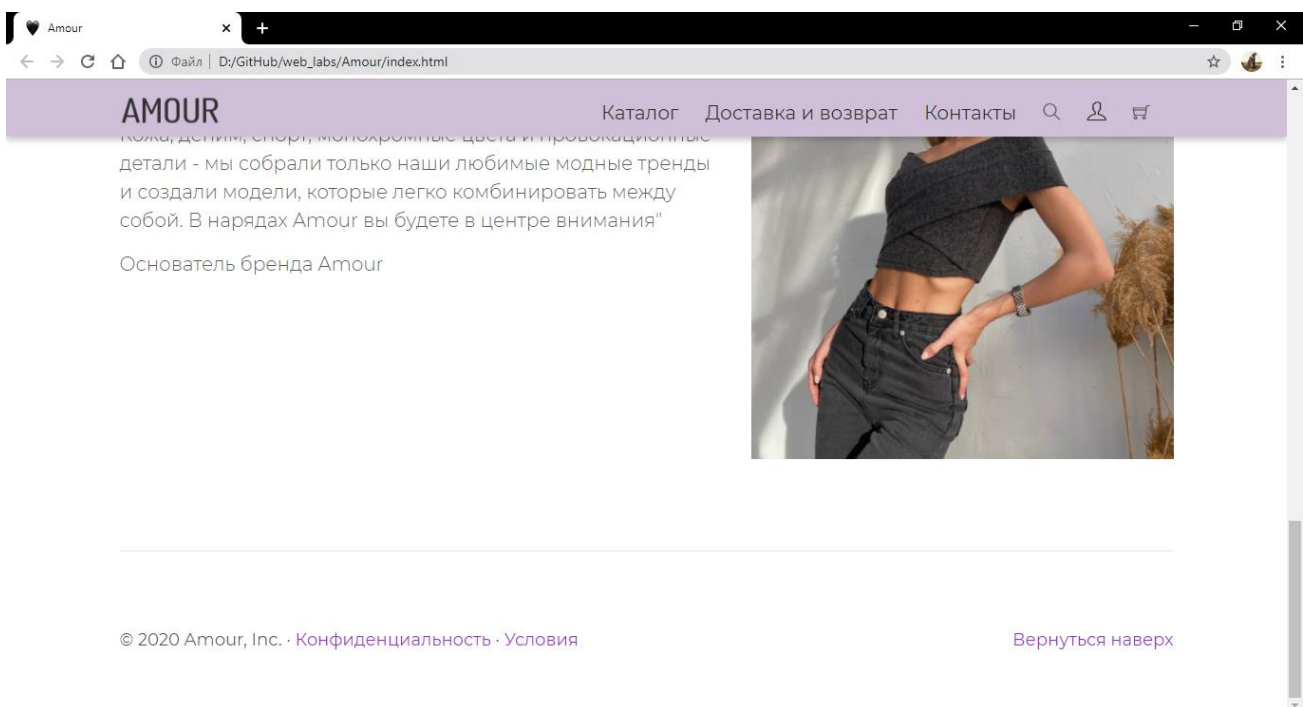
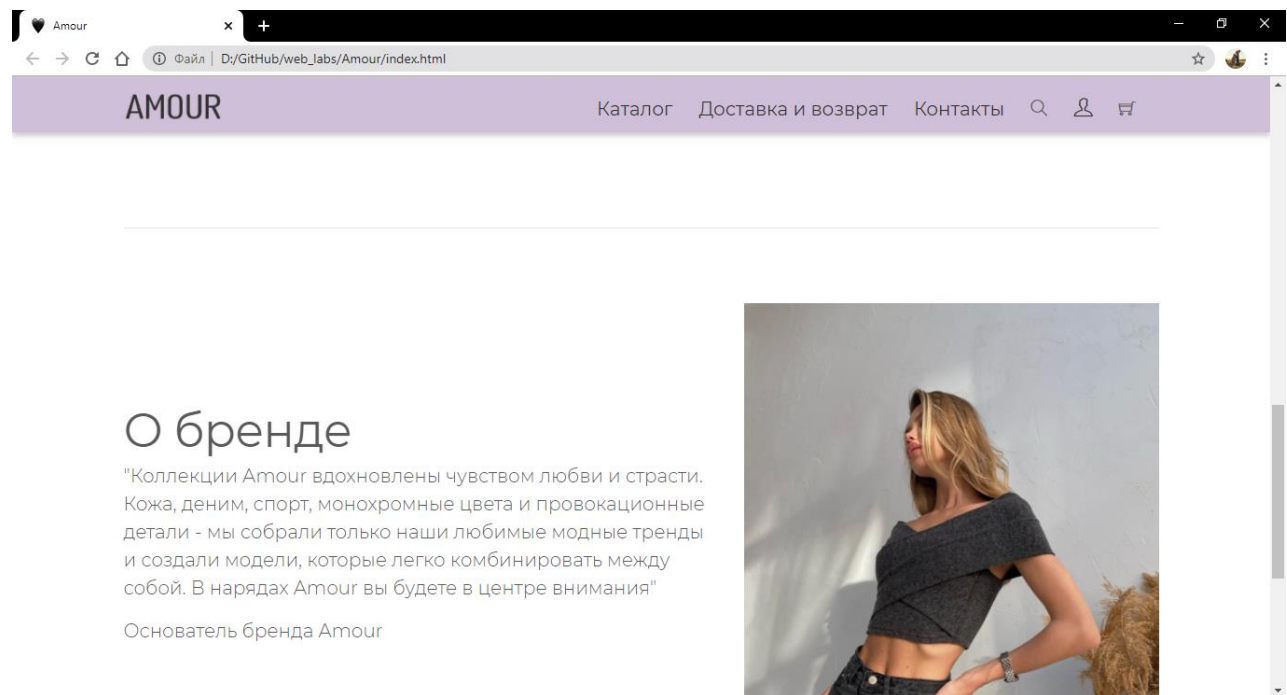
В качестве темы работы был выбран веб-сервис «Интернет-магазин одежды». Функционал включает в себя регистрацию и авторизацию пользователя, просмотр информации о товаре и покупку товара. Для Разработки frontend-приложения был выбран фреймворк для JavaScript - Vue JS, для backend – Yii2 на php, а в качестве СУБД выбрана MySQL.

1. Разработка шаблонов страниц

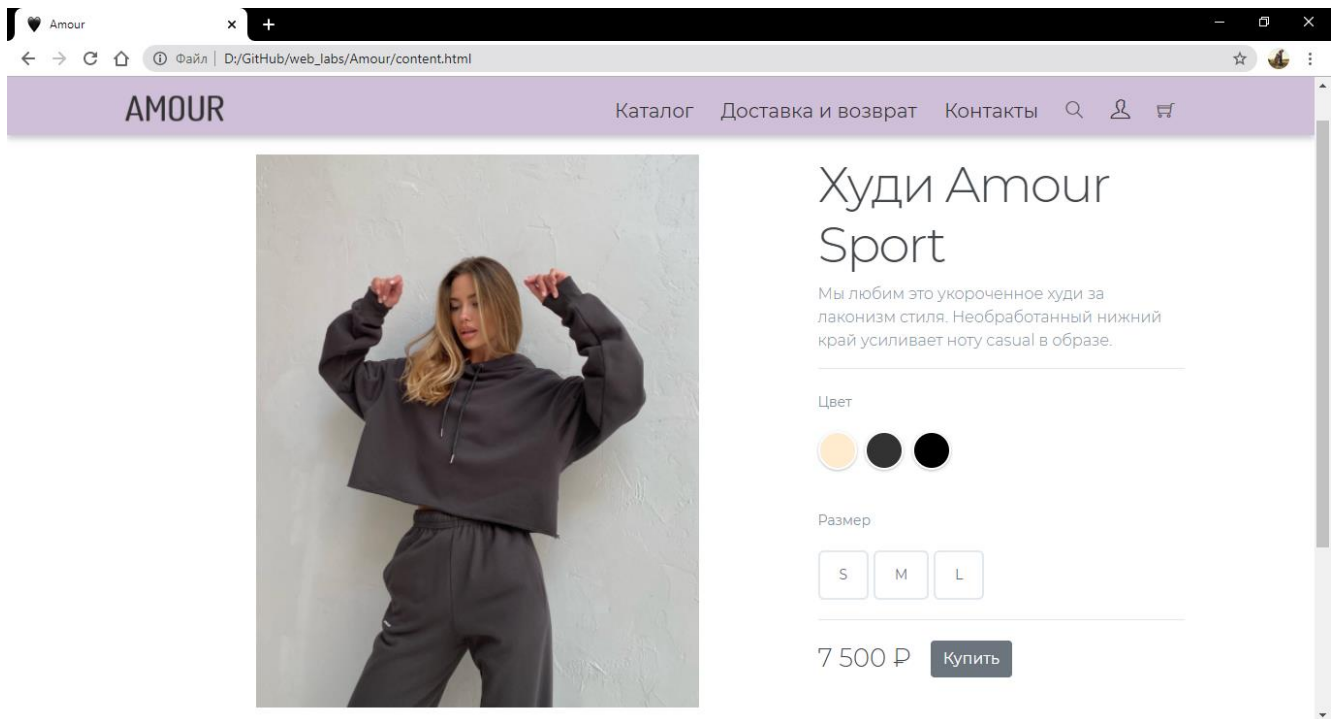
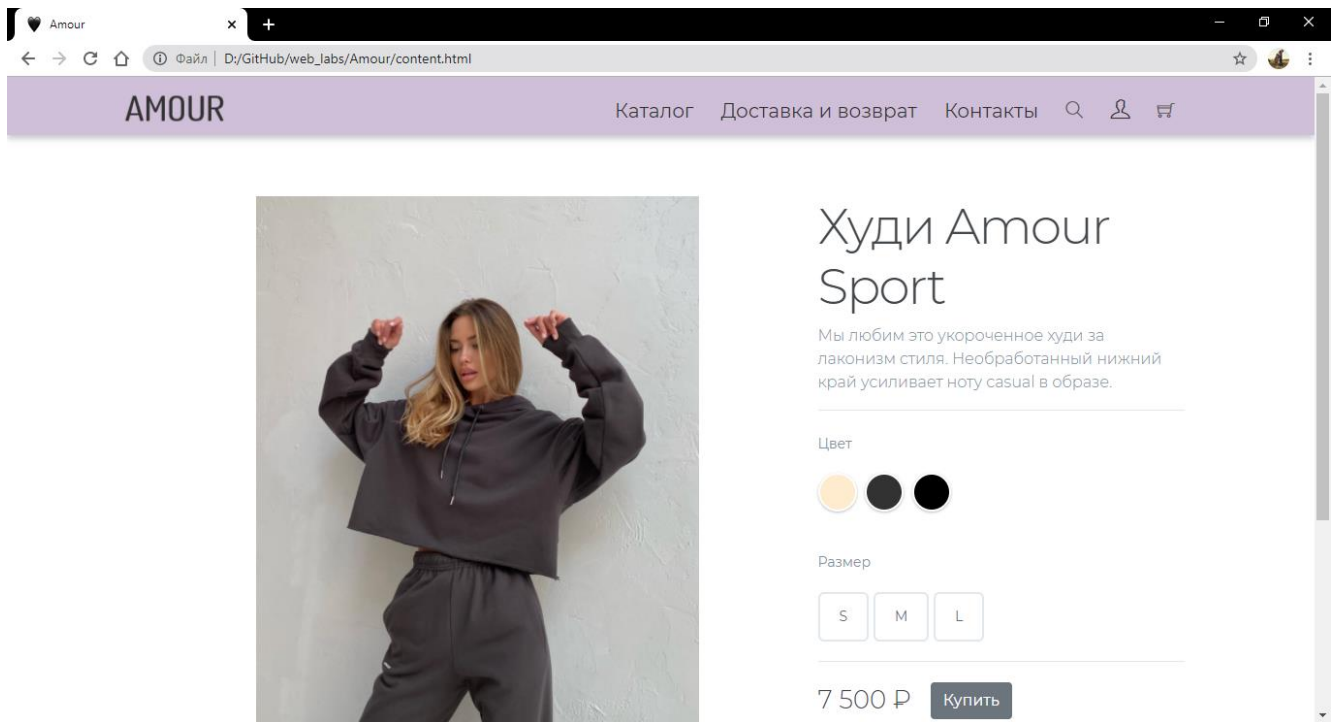
Разработаны начальные макеты страниц: главная, контентная, авторизации и регистрации с использованием Bootstrap - фреймворка для HTML и CSS:

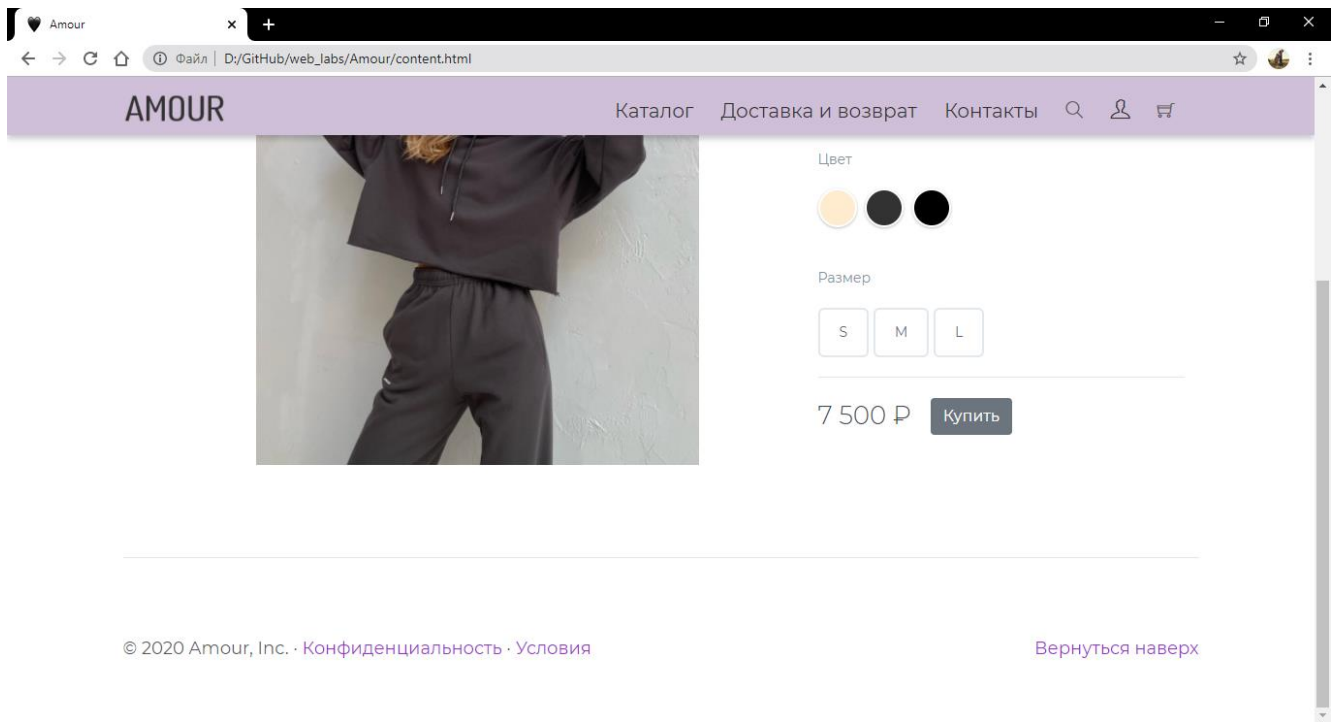
Главная страница:



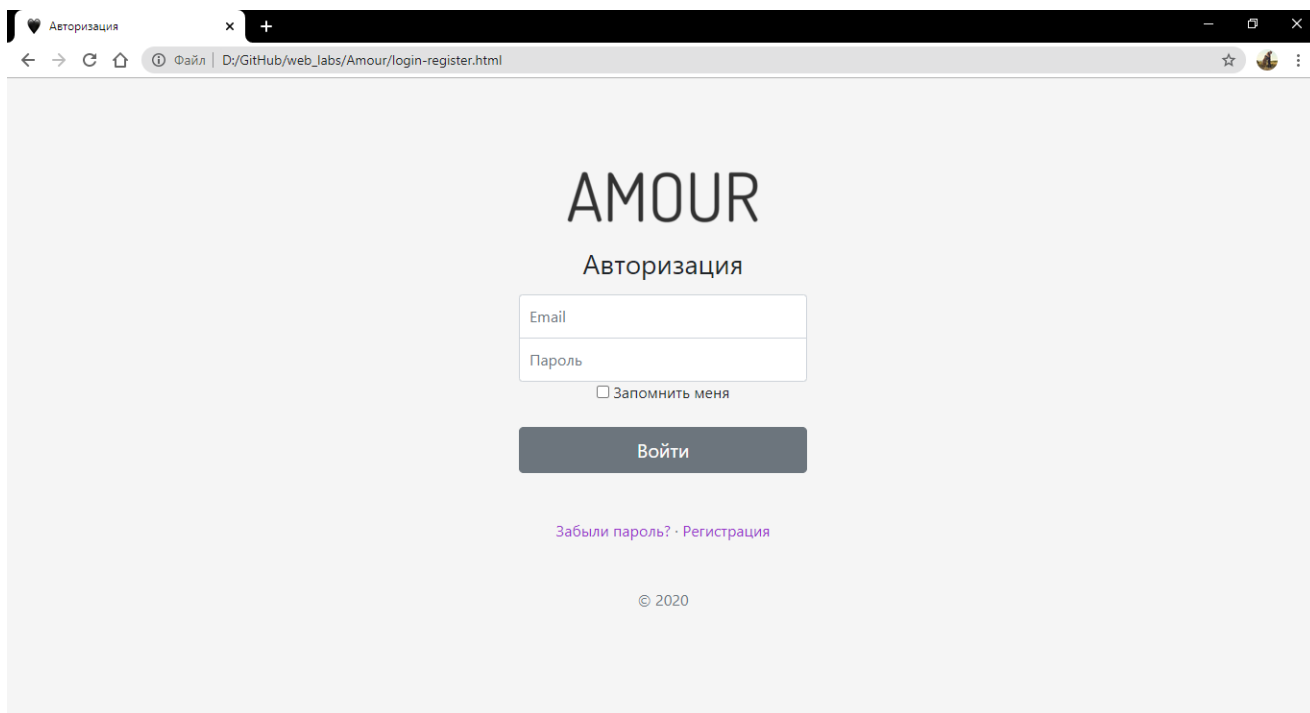


Контентная страница:





Страница авторизации:



A screenshot of a web browser displaying the login page for 'AMOUR'. The browser's address bar shows the file path 'D:/GitHub/web_labs/Amour/login-register.html'. The page has a light gray background. At the top center is the word 'AMOUR' in a large, dark, sans-serif font. Below it is the word 'Авторизация' in a smaller, dark font. The login form consists of two white input fields with thin gray borders: the first is labeled 'Email' and the second is labeled 'Пароль'. Below these fields is a checkbox with the text 'Запомнить меня'. A dark gray button with the text 'Войти' is centered below the checkbox. Below the button is a link that says 'Забыли пароль? · Регистрация' in a purple font. At the bottom center is the copyright notice '© 2020'.

AMOUR

Авторизация

Email

Пароль

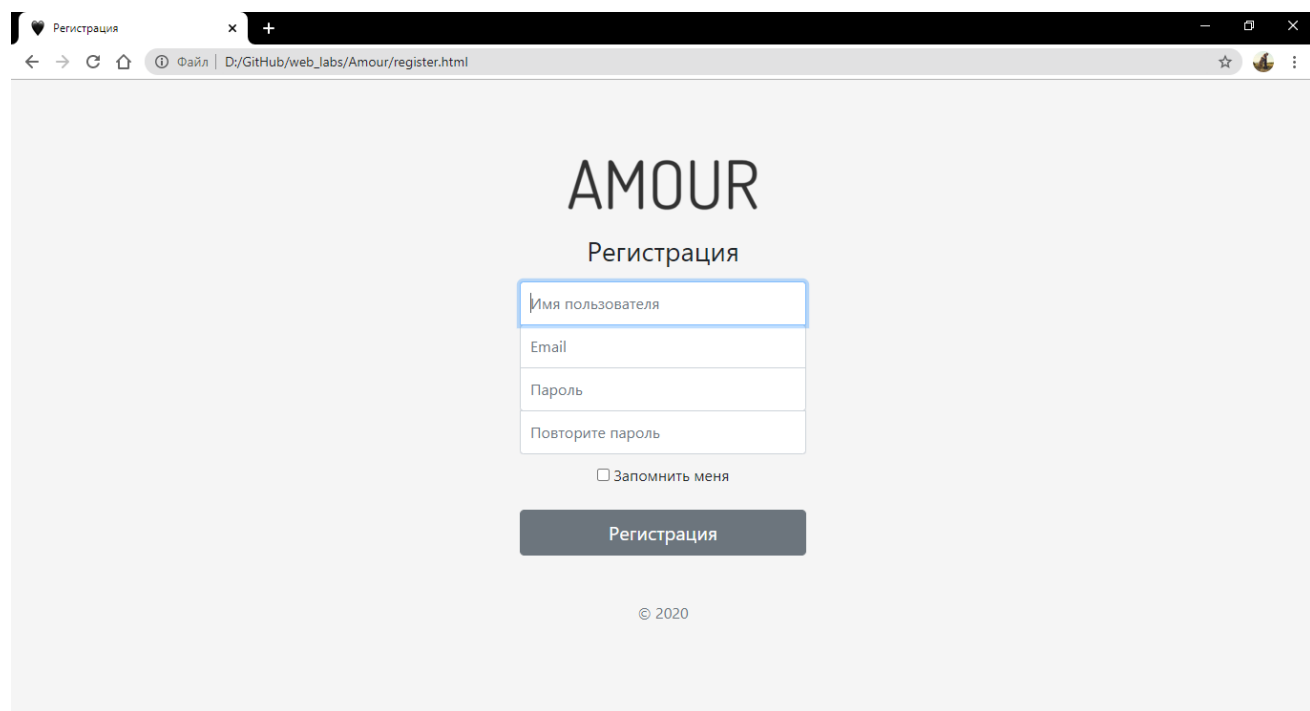
☐ Запомнить меня

Войти

[Забыли пароль? · Регистрация](#)

© 2020

Страница регистрации:



A screenshot of a web browser displaying the registration page for 'AMOUR'. The browser's address bar shows the file path 'D:/GitHub/web_labs/Amour/register.html'. The page has a light gray background. At the top center is the word 'AMOUR' in a large, dark, sans-serif font. Below it is the word 'Регистрация' in a smaller, dark font. The registration form consists of four white input fields with thin gray borders: the first is labeled 'Имя пользователя' and is currently active with a blue border; the second is labeled 'Email'; the third is labeled 'Пароль'; and the fourth is labeled 'Повторите пароль'. Below these fields is a checkbox with the text 'Запомнить меня'. A dark gray button with the text 'Регистрация' is centered below the checkbox. Below the button is the copyright notice '© 2020'.

AMOUR

Регистрация

Имя пользователя

Email

Пароль

Повторите пароль

☐ Запомнить меня

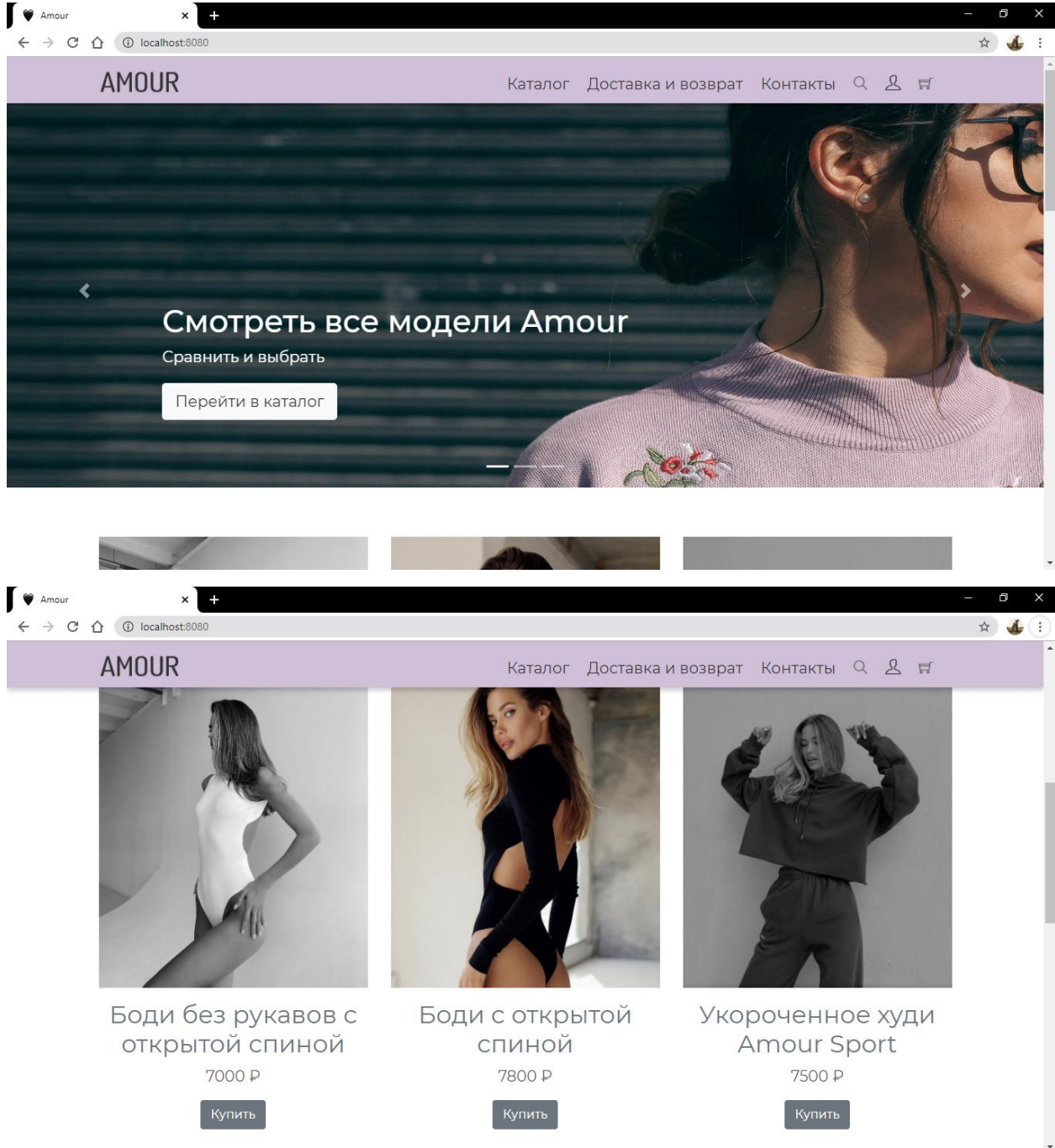
Регистрация

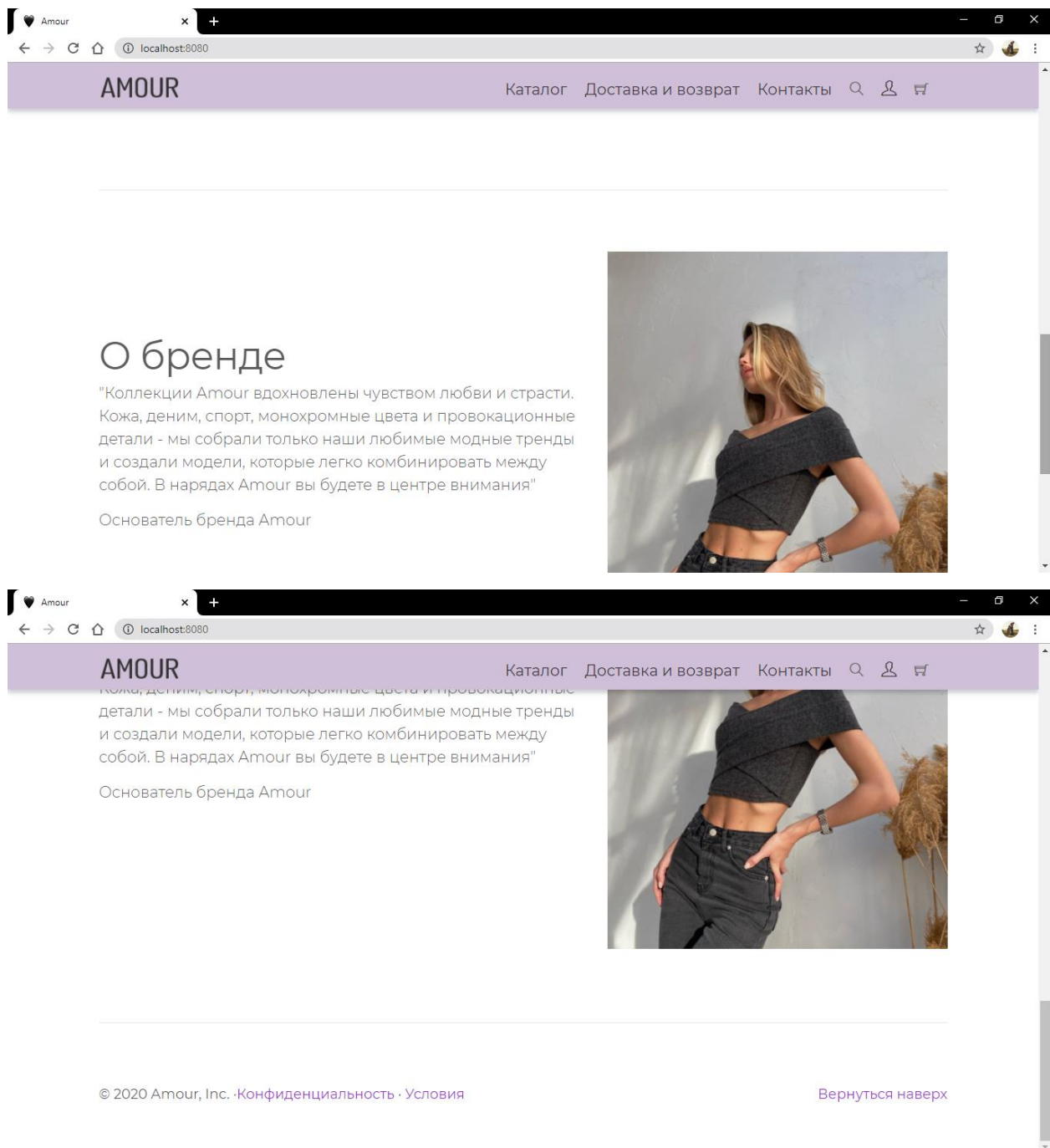
© 2020

2. Перенос макетов на Vue JS

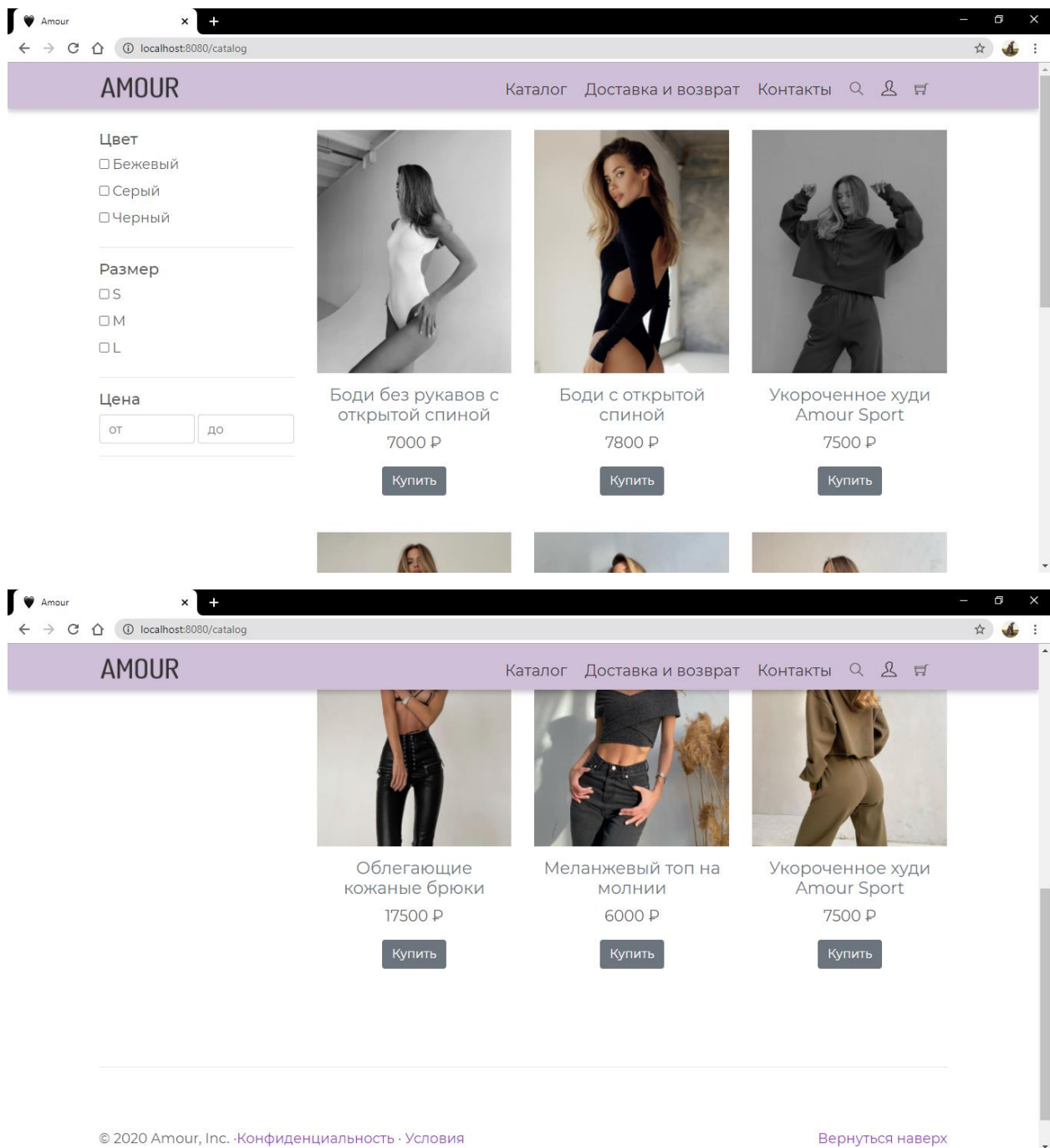
Макеты страниц из предыдущего пункта были доработаны и перенесены в компоненты фреймворка Vue JS. Также реализованы переходы между страницами, добавлены страницы пользователя и корзины.

Главная страница:

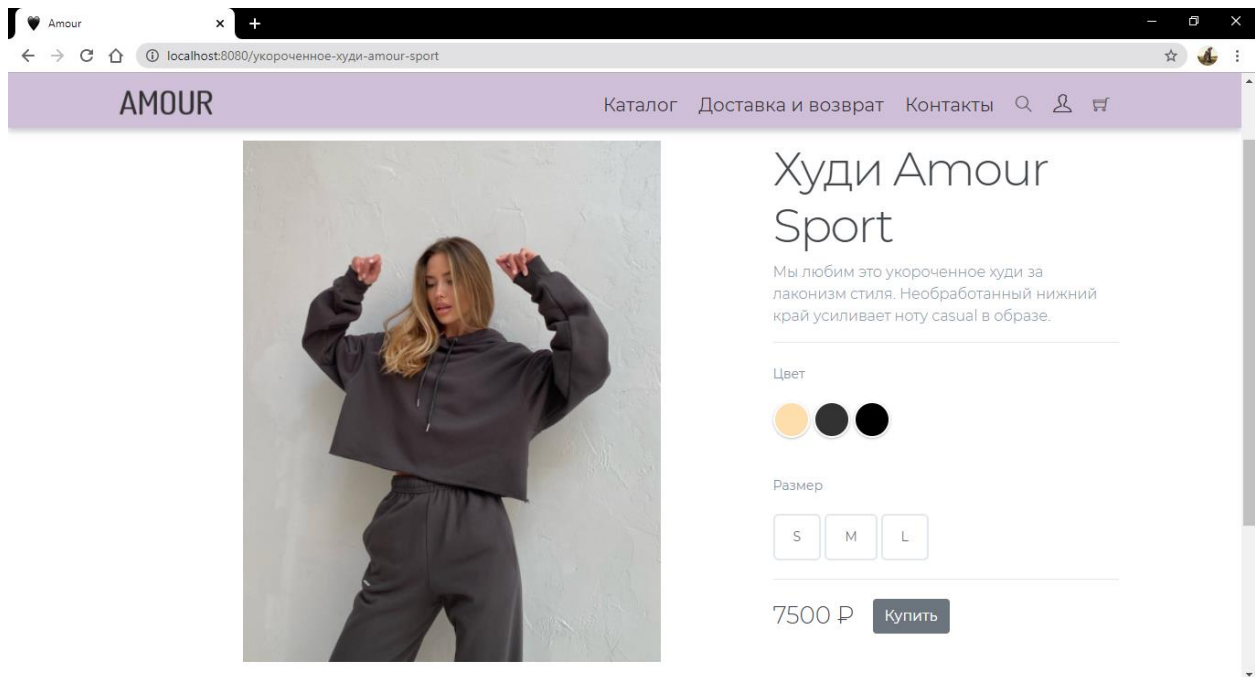




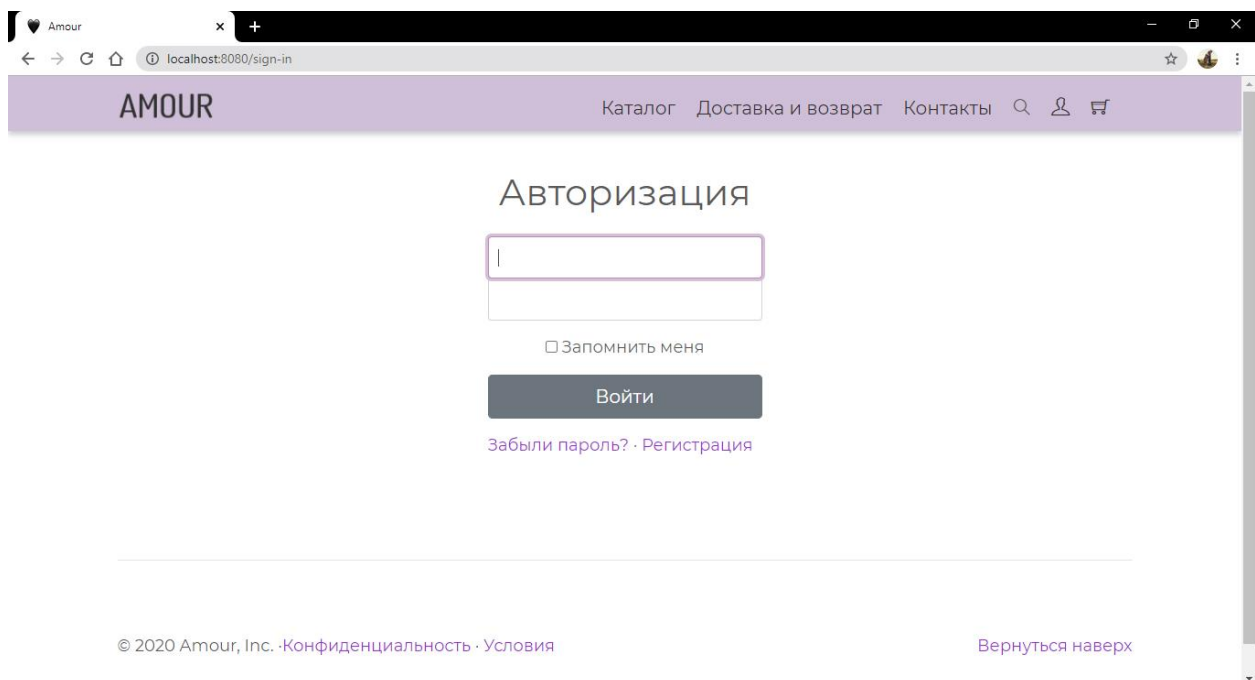
Каталог:



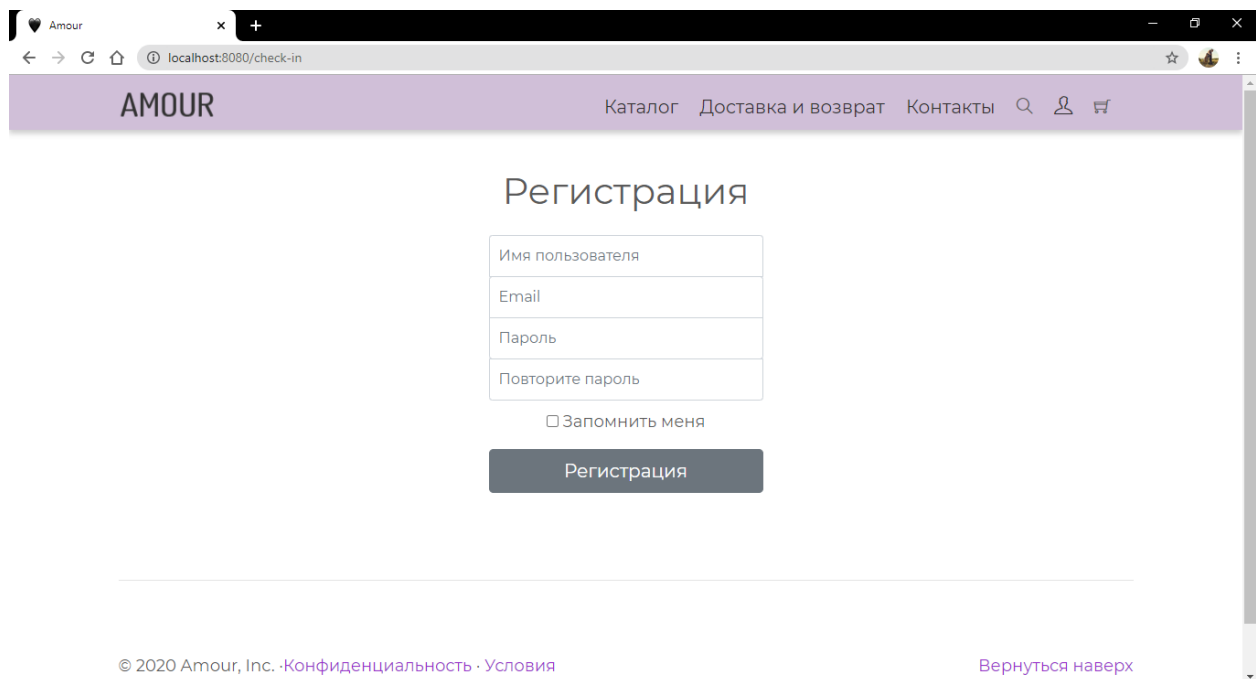
Контентная страница:



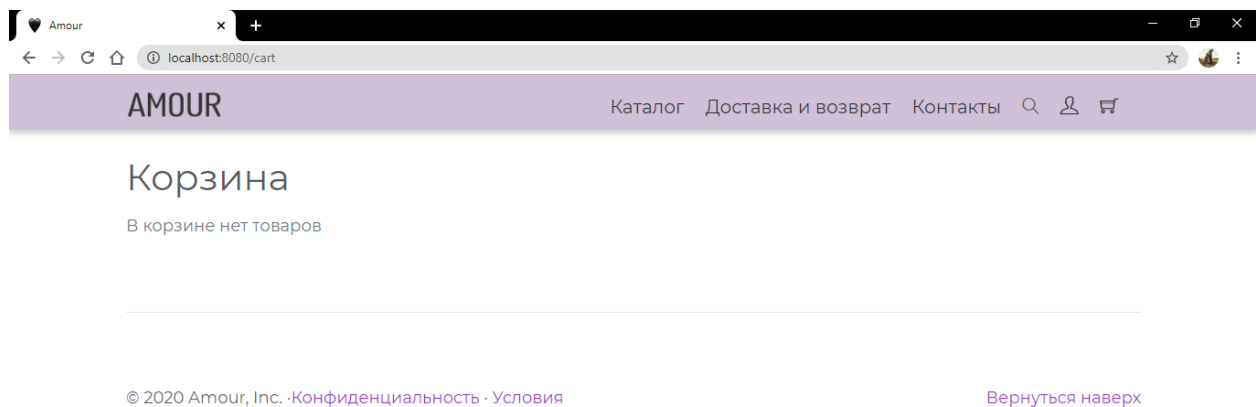
Страница авторизации:



Страница регистрации:



Страница корзины:




Amour

localhost:8080/cart

КаталогДоставка и возвратКонтакты7000 Р

Корзина

Товар	Стоимость	
 Боди без рукавов с открытой спиной	7000 Р	Удалить
Итого:	7000 Р	

© 2020 Amour, Inc. · [Конфиденциальность](#) · [Условия](#)

[Вернуться наверх](#)

Страница пользователя:

Amour

localhost:8080/profile

КаталогДоставка и возвратКонтактыvika@ya.ru

Виктория

vika@ya.ru

[Выйти](#)

© 2020 Amour, Inc. · [Конфиденциальность](#) · [Условия](#)

[Вернуться наверх](#)

3. Разворачивание базового приложения Yii2

Развернули базовое приложение Yii2 App Basic с помощью команды: “composer create-project --prefer-dist yiisoft/yii2-app-basic basic”. Затем это приложение было настроено для работы и дополнено результатами предыдущих лабораторных работ. Константные данные были перенесены в бэкенд сайта и вызываются посредством REST API.

Результаты работы запросов в Postman:

The screenshot shows the Postman interface with a GET request to `http://localhost:1199/v1/product/catalog` sent successfully. The response is a JSON array of two product objects.

Query Params

KEY	VALUE	DESCRIPTION
-----	-------	-------------

Body

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "name": "Боди без рукавов с открытой спиной",
5     "image": "images/t6.jpg",
6     "url": "боди-без-рукавов-с-открытой-спиной",
7     "price": 7000
8   },
9   {
10    "id": 2,
11    "name": "Боди с открытой спиной",
12    "image": "images/t2_1.jpg",
13    "url": "боди-с-открытой-спиной",
14    "price": 7800
15  },
16 ]
```

Status: 200 OK Time: 4.65 s Size: 1.18 KB

Build Browse

Launchpad

GET http://localhost:1199/v1/product/home

No Environment

Untitled Request

BUILD

GET

http://localhost:1199/v1/product/home

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE	DESCRIPTION

Body

Cookies

Headers (7)

Test Results

Status: 200 OK

Time: 3.26 s

Size: 739 B

Saved

Pretty

Raw

Preview

Visualize

JSON

```

1  {
2    "id": 1,
3    "name": "Боди без рукавов с открытой спиной",
4    "url": "боди-без-рукавов-с-открытой-спиной",
5    "image": "images/t6.jpg",
6    "price": 7000
7  },
8  {
9    "id": 2,
10   "name": "Боди с открытой спиной",
11   "url": "боди-с-открытой-спиной",
12   "image": "images/t2_1.jpg",
13   "price": 7800
14 },
15 ],
16 {
17   "id": 1,

```


Launchpad

GET http://localhost:1199/v1/page/c...

+...

No Environment

Untitled RequestBUILD

GET

http://localhost:1199/v1/page/contact

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

Query Params

KEY	VALUE	DESCRIPTION	...
-----	-------	-------------	-----

BodyCookiesHeaders (7)Test Results

Status: 200 OKTime: 4.75 sSize: 331 BSave

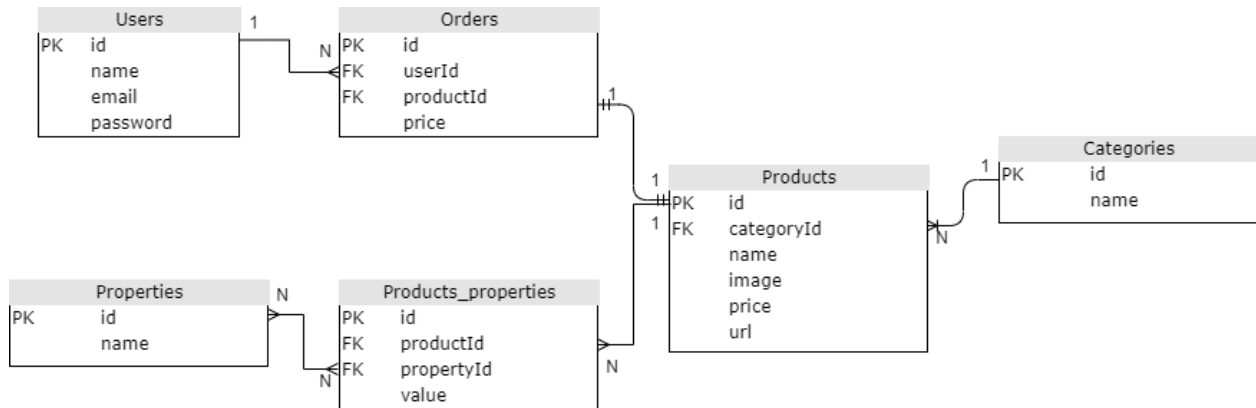
PrettyRawPreviewVisualize

JSON

```
1 {
2   "id": 2,
3   "name": "Контакты",
4   "text": "Информация отсутствует"
5 }
```

4. Проектирование и разработка базы данных

Спроектировали схему базы данных:



В качестве СУБД был выбран MySQL. Были созданы миграции для создания структуры базы данных:

```
PS D:\GitHub\web_labs\Amour> docker-compose run php yii migrate/create create_users_table
Creating amour_php_run ... done
usermod: no changes
Yii Migration Tool (based on Yii v2.0.39.3)
```

```
Create new migration '/app/migrations/m201229_125220_create_users_table.php'? (yes|no) [no]:yes
New migration created successfully.
PS D:\GitHub\web_labs\Amour> █
```

```
PS D:\GitHub\web_labs\Amour> docker-compose run php yii migrate
Creating amour_php_run ... done
usermod: no changes
Yii Migration Tool (based on Yii v2.0.39.3)
```

```
Total 1 new migration to be applied:
m201229_125220_create_users_table
```

```
Apply the above migration? (yes|no) [no]:yes
*** applying m201229_125220_create_users_table
> create table {%users%} ... done (time: 0.388s)
*** applied m201229_125220_create_users_table (time: 0.606s)
```

```
1 migration was applied.
```

```
Migrated up successfully.
```

```
PS D:\GitHub\web_labs\Amour> █
```

Полученная структура БД:

Таблица: products

[Выбрать](#)
[Показать структуру](#)
[Изменить таблицу](#)
[Новая запись](#)

поле	Тип	Комментарий
id	int <i>Автоматическое приращение</i>	
name	varchar(128)	Название
image	varchar(128) <i>NULL</i>	Изображение
price	decimal(12,2) <i>NULL [0.00]</i>	Цена
createdAt	datetime	Дата создания
updatedAt	datetime <i>NULL</i>	Дата изменения
url	varchar(128)	URL
categoryId	int <i>NULL</i>	Категория

Индексы

PRIMARY	<i>id</i>
INDEX	<i>categoryId</i>

[Изменить индексы](#)

Внешние ключи

Источник	Цель	При стирании	При обновлении	
categoryId	<i>categories(id)</i>	SET NULL	RESTRICT	Изменить

Таблица: categories

[Выбрать](#)
[Показать структуру](#)
[Изменить таблицу](#)
[Новая запись](#)

поле	Тип	Комментарий
id	int <i>Автоматическое приращение</i>	
name	varchar(64)	Наименование
createdAt	datetime	Дата создания
updatedAt	datetime <i>NULL</i>	Дата изменения

Индексы

PRIMARY	<i>id</i>
----------------	-----------

Таблица: properties

[Выбрать](#)
[Показать структуру](#)
[Изменить таблицу](#)
[Новая запись](#)

поле	Тип	Комментарий
id	int <i>Автоматическое приращение</i>	
name	varchar(64)	Наименование
createdAt	datetime	Дата создания
updatedAt	datetime <i>NULL</i>	Дата изменения

Индексы

PRIMARY	<i>id</i>
----------------	-----------

Таблица: products_properties

[Выбрать](#)
[Показать структуру](#)
[Изменить таблицу](#)
[Новая запись](#)

поле	Тип	Комментарий
id	int <i>Автоматическое приращение</i>	
productId	int	Товар
propertyId	int	Свойство
value	text	Значение
createdAt	datetime	Дата создания
updatedAt	datetime <i>NULL</i>	Дата изменения

Индексы

PRIMARY	<i>id</i>
INDEX	<i>productId</i>
INDEX	<i>propertyId</i>

[Изменить индексы](#)

Внешние ключи

Источник	Цель	При стирании	При обновлении	
productId	products(<i>id</i>)	CASCADE	RESTRICT	Изменить
propertyId	properties(<i>id</i>)	CASCADE	RESTRICT	Изменить

Таблица: users

[Выбрать](#)
[Показать структуру](#)
[Изменить таблицу](#)
[Новая запись](#)

поле	Тип	Комментарий
id	int <i>Автоматическое приращение</i>	
name	varchar(128)	Имя пользователя
email	varchar(100) <i>NULL</i>	Электронная почта
password	varchar(50)	Пароль
createdAt	datetime <i>NULL</i>	Дата создания
updatedAt	datetime <i>NULL</i>	Дата изменения

Индексы

PRIMARY	<i>id</i>
----------------	-----------

Таблица: orders

[Выбрать](#)
[Показать структуру](#)
[Изменить таблицу](#)
[Новая запись](#)

поле	Тип	Комментарий
id	int <i>Автоматическое приращение</i>	
productId	int	Продукт
userId	int	Пользователь
price	float <i>NULL</i>	Сумма заказа
createdAt	datetime <i>NULL</i>	Дата создания
updatedAt	datetime <i>NULL</i>	Дата изменения

Индексы

PRIMARY	<i>id</i>
INDEX	<i>productId</i>
INDEX	<i>userId</i>

[Изменить индексы](#)

Внешние ключи

Источник	Цель	При стирании	При обновлении	
productId	<i>products(id)</i>	CASCADE	RESTRICT	Изменить
userId	<i>users(id)</i>	CASCADE	RESTRICT	Изменить

Созданные модели, соответствующие сущностям БД:

```

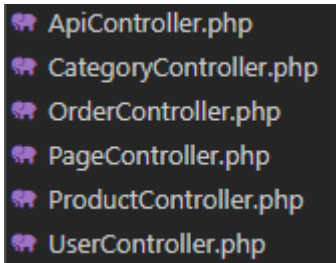
🐘 BaseModel.php
🐘 Category.php
🐘 LoginForm.php
🐘 Order.php
🐘 Product.php
🐘 ProductProperty.php
🐘 Property.php
🐘 User.php

```

5. Разработка REST API

Разработали REST API, документацию к нему, а также дополнили функционал некоторых моделей БД.

Разработанные контроллеры:



Документация к API:

Общее описание к методам:

Базовый URL: <http://192.168.0.195::1199/v1>

Контроллеры:

1. Контроллер категорий товаров CategoryController

URL: /category

Тип запроса: GET

Действия:

Название/url	Описание	Параметры		
		параметр	тип	обязательный
actionList	Возвращает все категории	Нет		
/list				

2. Контроллер заказов OrderController

URL: /order

Тип запроса: POST

Действия:

Название/url	Описание	Данные
actionCreate	Создаёт заказ с привязкой к пользователю и продукту по их id	{userId, productId, price}
/create		

Тип запроса: POST

Действия:

Название/url	Описание	Данные
actionDelete	Удаляет заказ по id пользователя и id продукта	{userId, productId, price}
/delete		

3. Контроллер страниц PageController

URL: /page

Тип запроса: GET

Действия:

Название/url	Описание	Параметры		
		параметр	тип	обязательный
actionDelivery /delivery	Возвращает информацию о странице «Доставка»	Нет		
actionContact /contact	Возвращает информацию о странице «Контакты»	Нет		

4. Контроллер товаров ProductController

URL: /product

Тип запроса: GET

Действия:

Название/url	Описание	Параметры		
		параметр	тип	обязательный
actionHome /home	Возвращает 3 последних добавленных товара	Нет		
actionCatalog /catalog	Возвращает все товары	Нет		
actionInfo /info	Возвращает информацию о товаре	url	String(128)	Да

5. Контроллер пользователей UserController

URL: /user

Тип запроса: GET

Действия:

Название/url	Описание	Параметры		
		параметр	тип	обязательный
actionLogin /login	Выполняет авторизацию пользователя по его email и password. Возвращает соответствующего пользователя или исключение с кодом 406 – неверный логин или пароль	email password	String(100) String(50)	Да Да

Тип запроса: POST

Действия:

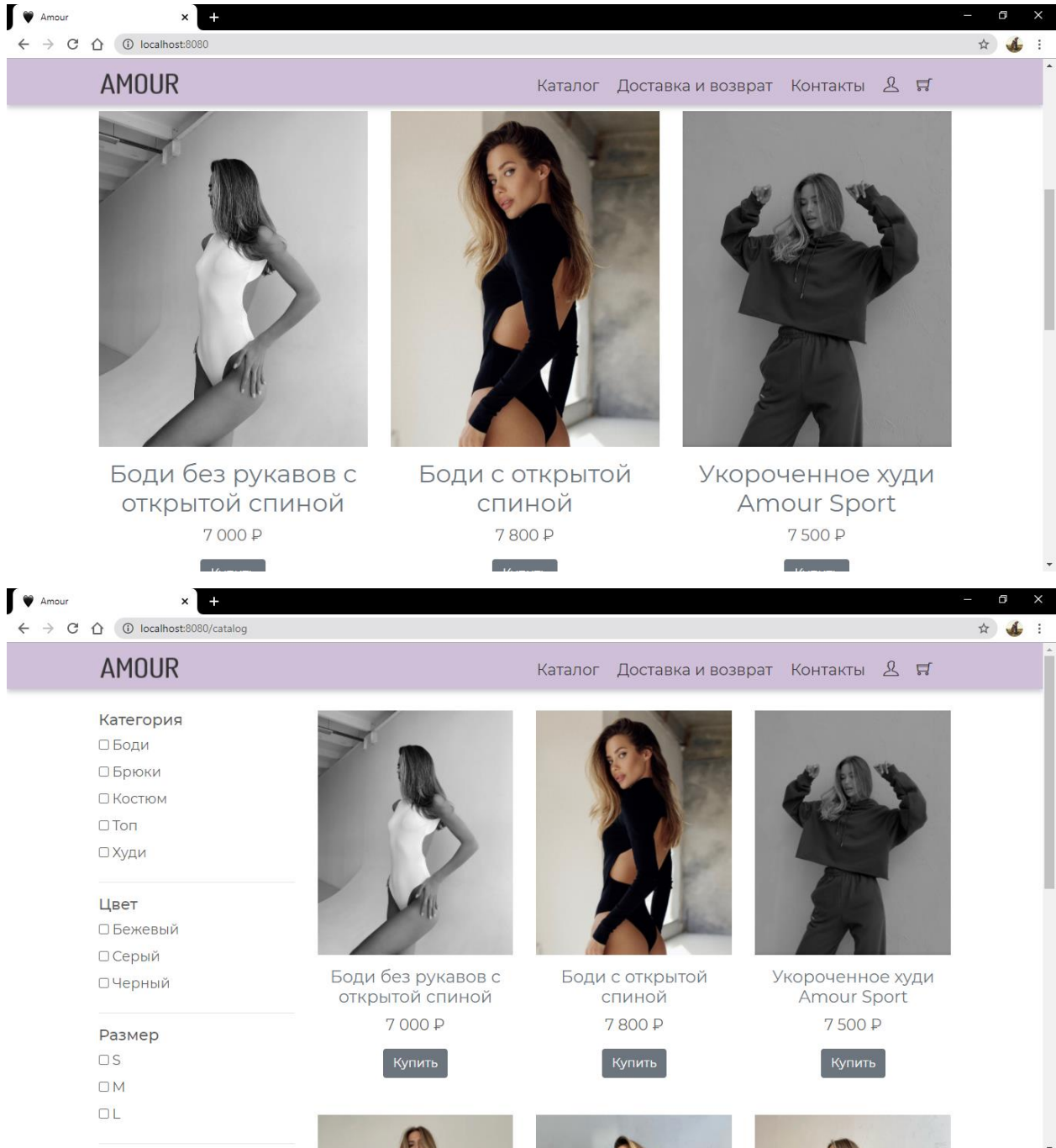
Название/url	Описание	Данные
actionRegistration /registration	Создаёт пользователя с указанным email, name и password, если пользователя с таким email не существует, иначе – исключение с кодом 406	{email, name, password}

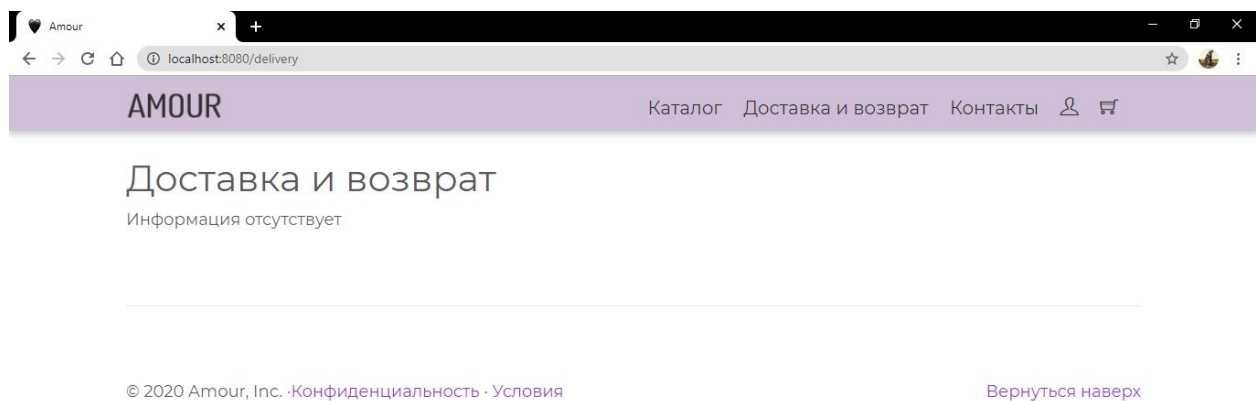
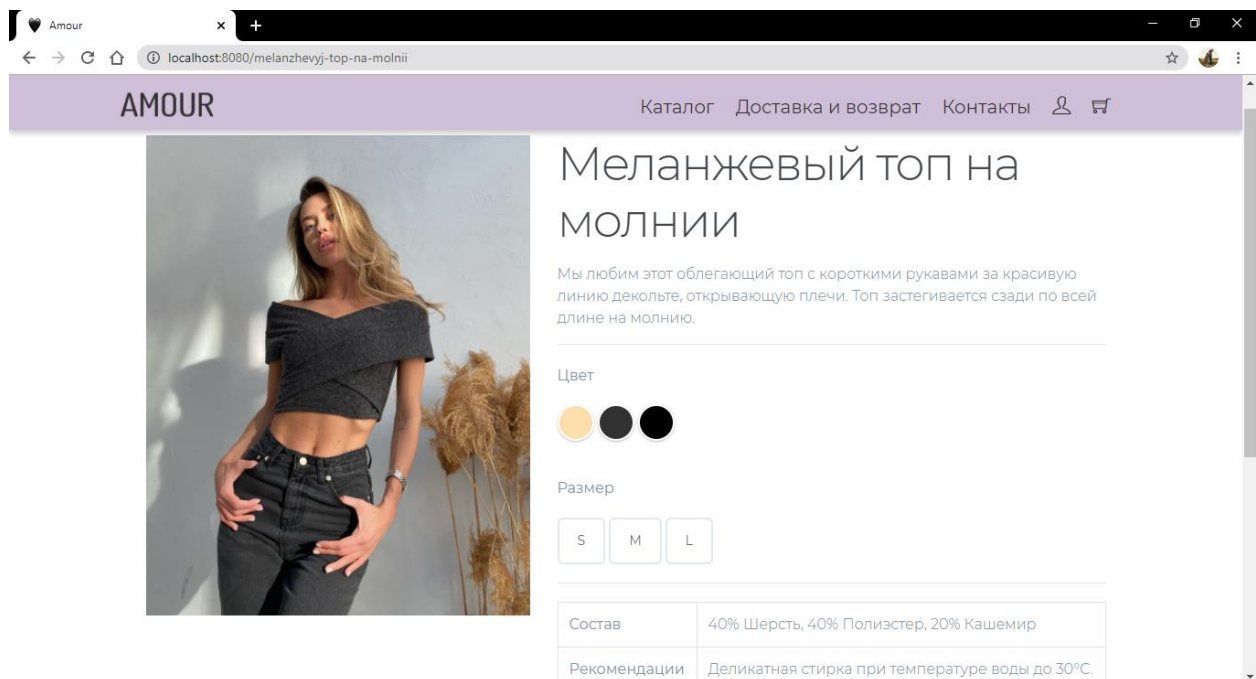
6. Работа с HTTP запросами

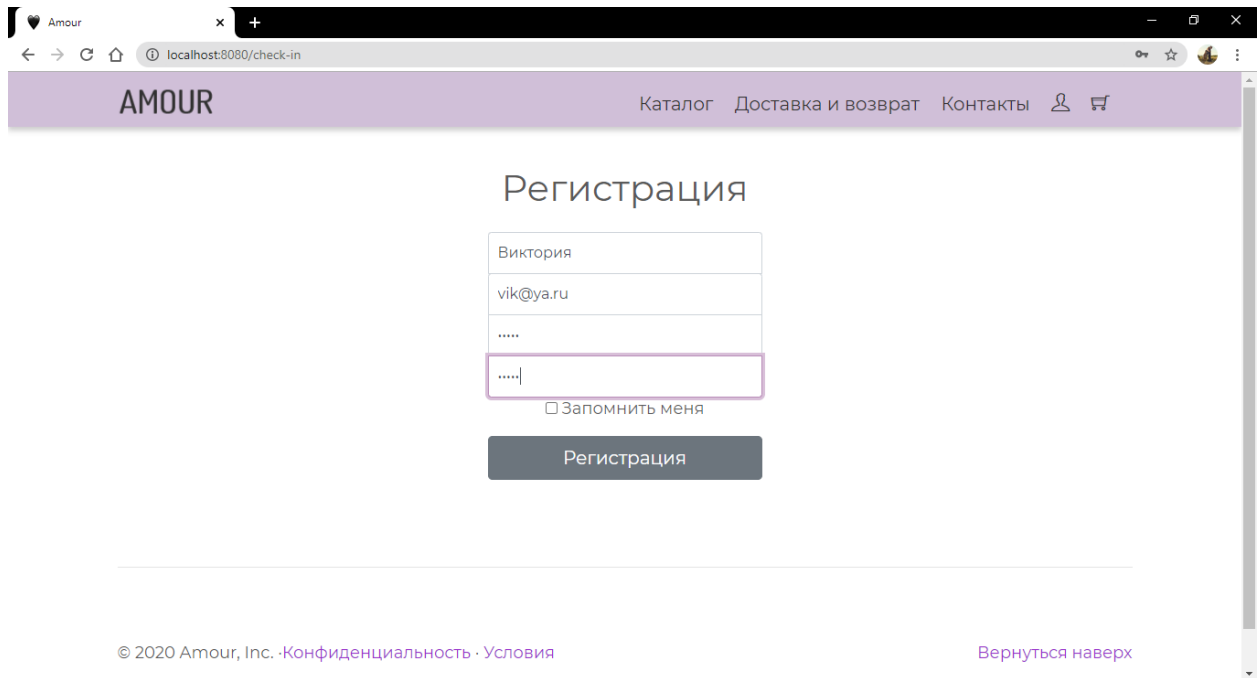
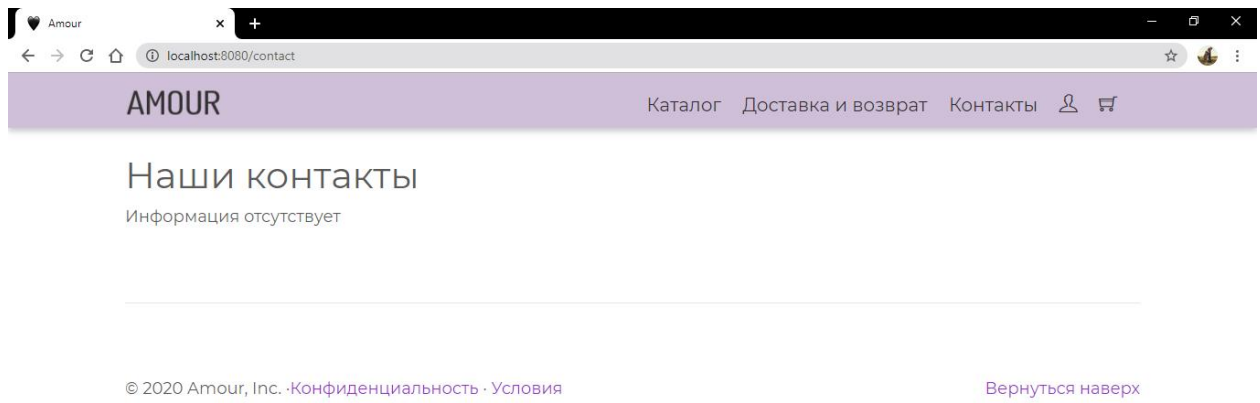
Для работы с AJAX запросами была выбрана библиотека Axios, представляющая собой HTTP-клиент, основанный на промисах и предназначенный для браузеров и Node.js.

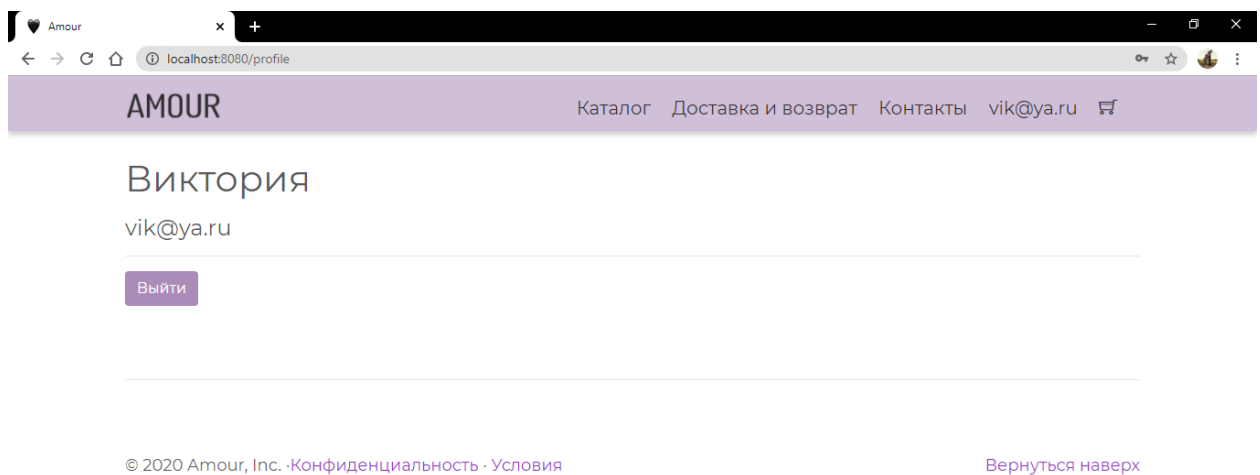
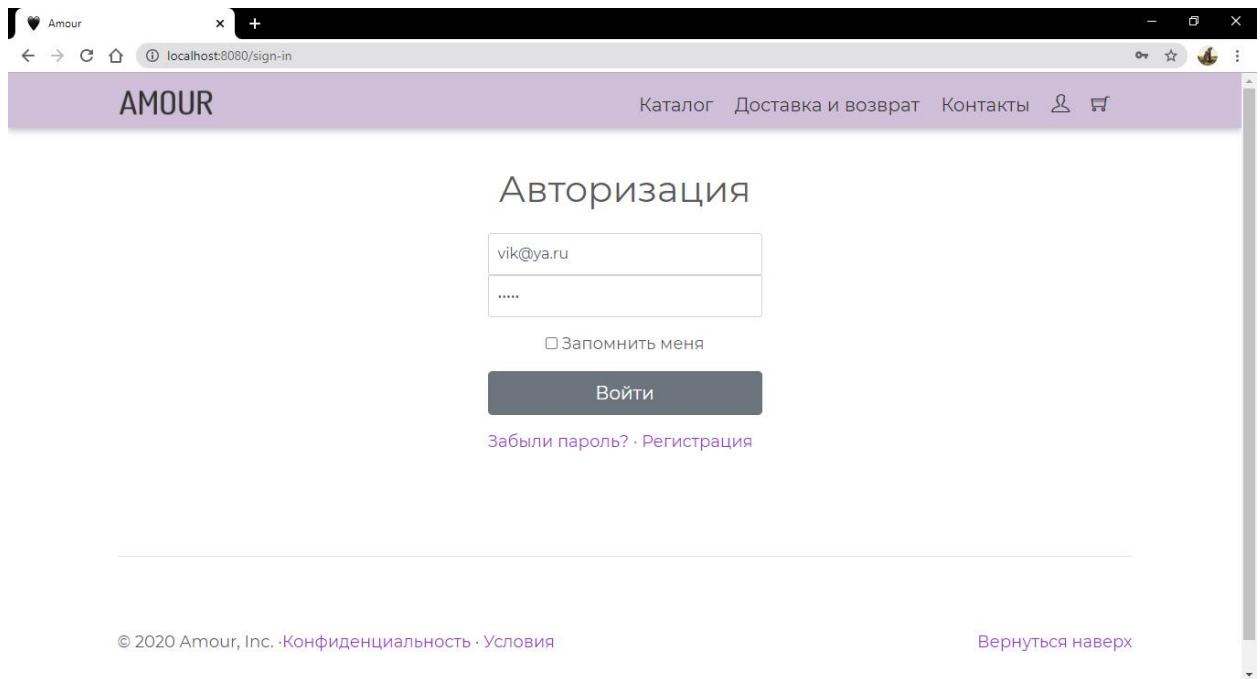
В ходе работы было реализовано взаимодействие фронтенда с REST API с помощью кроссдоменных HTTP запросов.

Полученный сайт:










Amour

localhost:8080/cart

КаталогДоставка и возвратКонтакты7500 Р

Корзина

Товар	Стоимость	
 Укороченное худи Amour Sport	7 500 Р	Удалить
Итого:	7500 Р	

© 2020 Amour, Inc. · [Конфиденциальность](#) · [Условия](#)

[Вернуться наверх](#)

Вывод

В результате выполнения работы были изучены основы работы протокола HTTP, языков HTML и CSS, а также получили практические навыки в программировании на языках программирования JavaScript и PHP. Реализовали frontend-приложение на основе макетов страниц на HTML+CSS и скриптов на JS с помощью фреймворка Vue JS, backend-приложение разработали на языке PHP с помощью Yii2. Базу данных реализовали с помощью миграций в СУБД MySQL, при этом backend-сервер и БД разместили в docker.

Приложение

Пример кода компонента Vue:

```
<template>
  <div class="products text-center mb-5">
    <div class="form-group">
      <router-link
        v-bind:to="{ name: 'Product', params: { url: product.url } }"
      >
        
      </router-link>
    </div>
    <h2>
      <router-link
        v-bind:to="{ name: 'Product', params: { url: product.url } }"
        class="text-secondary"
      >{{ product.name }}</router-link>
    </h2>
    <p>
      <span class="price">{{ product.formattedPrice }}</span>
    </p>
    <p>
      <button class="btn btn-secondary" v-on:click="addToCart">Купить</button>
    </p>
  </div>
</template>

<script>
import CartData from "@components/cart/cart";
import User from "@components/user/user";
import Order from "@components/cart/order";
import router from "@router";
import Axios from "axios";

export default {
  name: "Product",
  props: ["product"],
  data() {
    return {
      userId: 0,
      productId: 0,
      price: 0,
    };
  },
  methods: {
    addToCart() {
      CartData.add(this.product);
      if (User.isAuthenticated()) {
        this.$http
          .post("/order/create", {
            userId: User.id,
            productId: this.product.id,
            price: this.product.price
          })
          .then((response) => {
            Order.add(response.data);
          })
      }
    }
  }
}
```

```

        .catch((errors) => {console.log(errors)});
    },
},
};
</script>

```

Пример кода миграции:

```

<?php

use yii\db\Migration;

/**
 * Class m201228_134726_create_products_properties
 */
class m201228_134726_create_products_properties extends Migration
{
    /**
     * {@inheritdoc}
     */
    public function safeUp()
    {
        $this->createTable('{{%products_properties}}', [
            'id' => $this->primaryKey(),
            'productId' => $this->integer()->notNull()->comment('Товар'),
            'propertyId' => $this->integer()->notNull()->comment('Свойство'),
            'value' => $this->text()->notNull()->comment('Значение'),
            'createdAt' => $this->dateTime()->notNull()->comment('Дата создания'),
            'updatedAt' => $this->dateTime()->comment('Дата изменения')
        ]);

        $this->addForeignKey('fk_products_properties_productId', '{{%products_properties}}', 'productId', '{{%products}}', 'id', 'CASCADE');
        $this->addForeignKey('fk_products_properties_propertyId', '{{%products_properties}}', 'propertyId', '{{%properties}}', 'id', 'CASCADE');
    }

    /**
     * {@inheritdoc}
     */
    public function safeDown()
    {
        $this->dropForeignKey('fk_products_properties_productId', '{{%products_properties}}');
        $this->dropForeignKey('fk_products_properties_propertyId', '{{%products_properties}}');
        $this->dropTable('{{%products_properties}}');
    }
}

```

Пример кода модели:

```

<?php

namespace app\modules\v1\models;
use yii\behaviors\SluggableBehavior;

```

```

use Yii;

/**
 * This is the model class for table "products".
 *
 * @property int $id
 * @property string $name Название
 * @property string|null $image Изображение
 * @property float|null $price Цена
 * @property string $createdAt Дата создания
 * @property string|null $updatedAt Дата изменения
 * @property string $url URL
 * @property string|null $categoryId Категория
 *
 * @property Category $category
 * @property Property[] $properties
 * @property ProductProperty[] $propertiesValues
 */
class Product extends BaseModel
{
    /**
     * {@inheritdoc}
     */
    public static function tableName()
    {
        return 'products';
    }

    /**
     * {@inheritdoc}
     */
    public function rules()
    {
        return [
            [['name'], 'required'],
            [['price'], 'number'],
            [['categoryId'], 'integer'],
            [['createdAt', 'updatedAt'], 'safe'],
            [['name', 'image', 'url'], 'string', 'max' => 128],
        ];
    }

    /**
     * {@inheritdoc}
     */
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'name' => 'Название',
            'image' => 'Изображение',
            'price' => 'Цена',
            'createdAt' => 'Дата создания',
            'updatedAt' => 'Дата изменения',
            'url' => 'URL',
            'categoryId' => 'Категория'
        ];
    }
}

```



```

    }

    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        $behaviors = parent::behaviors();
        $behaviors[] = [
            'class' => SluggableBehavior::class,
            'attribute' => 'name',
            'slugAttribute' => 'url'
        ];

        return $behaviors;
    }

    /**
     * {@inheritdoc}
     */
    public function toArray(array $fields = [], array $expand = [], $recursive = true
)
    {
        return [
            'id' => $this->id,
            'name' => $this->name,
            'image' => $this->image,
            'price' => (double)$this->price,
            'formattedPrice' => number_format($this->price, 0, '.', ' ') . ' P',
            'url' => $this->url,
            'createdAt' => $this->createdAt,
            'category' => $this->category,
            'propertiesValues' => $this->propertiesValues,
            //'orders' => $this->orders
        ];
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getCategory()
    {
        return $this->hasOne(Category::class, ['id' => 'categoryId']);
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getProperties()
    {
        return $this->hasMany(Property::class, ['id' => 'property_id'])
            ->viaTable(ProductProperty::tableName(), ['product_id' => 'id']);
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getPropertiesValues()

```

```

{
    return $this->hasMany(ProductProperty::class, ['productId' => 'id']);
}

/**
 * @return \yii\db\ActiveQuery
 */
public function getOrders()
{
    return $this->hasMany(Order::class, ['productId' => 'id']);
}
}

```

Пример кода контроллера:

```

<?php

namespace app\modules\v1\controllers;

use app\modules\v1\models\LoginForm;
use app\modules\v1\models\User;
// use yii\db\ActiveRecord;
use yii\rest\Controller;
use yii\db\Exception;
use yii\web\NotAcceptableHttpException;
use Yii;

class UserController extends ApiController
{
    public function actionLogin($email, $password)
    {
        $form = User::find()
            ->where(['email' => $email])
            ->one();
        if ($form != null or $form != ''){
            if ($form->checkPass($password))
                return $form->getUser();
        }
        throw new NotAcceptableHttpException('Неверный логин или пароль!'); //406
    }

    public function actionRegistration() {
        $data = \Yii::$app->request->getBodyParams();
        if (User::find()
            ->where(['email' => $data['email']])
            ->one() != null){
            throw new NotAcceptableHttpException('Email занят!'); //406
            //return null;
        }

        $user = new User();
        $user->load($data, '');
        $user->save();
        return $user;
    }
}

```