# ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

При разработке системы используется объектно-ориентированный и функциональный подход программирования. Чтобы понимать технические аспекты системы, следует описать составляющие её классы и методы. Диаграмма классов приведена на чертеже ГУИР.400201.089 PP.1.

## 3.1 Блок обработки запросов клиента

## 3.1.1 Класс ProductListPageController

Класс ProductListPageController представляет из себя точку входа запроса клиента на список товаров, представленных в магазине.

Класс ProductListPageController содержит следующие поля:

* phoneServiceImpl: PhoneService – экземпляр класса PhoneService из блока сервис, отвечающий за работу с мобильными устройствами;
* cartService : CartService – экземпляр класса CartService из блока сервис, отвечающий за управление корзиной покупателей;
* httpSession: HttpSession – экземпляр класса HttpSession, который предоставляет инструменты для работы с HTTP-сессией.

Класс ProductListPageController содержит метод:

* showProductList(String, String, String, Long, Model) : String – принимает запрос от пользователя для отображения списка товаров, управляет сортировкой элементов (по различным полям и направлениям), задаёт параметры для поиска.

**3.1.2 Класс AjaxCartController**

Класс AjaxCartController представляет из себя обработчик запросов клиента на добавление товаров в корзину.

Класс AjaxCartController содержит следующие поля:

* quantityValidator: PhoneDtoValidator – экземпляр класса PhoneDtoValidator из блока валидации запросов, отвечающий за валидацию запроса;
* quantityValidator: CartService – экземпляр класса CartServiceImpl из блока сервис, отвечающий за управление корзиной покупателей;
* httpSession: HttpSession – экземпляр класса HttpSession, который предоставляет инструменты для работы с HTTP-сессией.

Класс AjaxCartController содержит метод:

* addPhone(PhoneDto, BindingResult) : ResponseEntity<?> – принимает запрос от пользователя для добавления товара в корзину, вызывает функции для валидации, отправляет ответ, сигнализирующий об результате действия.

## 3.1.3 Класс CartPageController

Класс CartPageController представляет из себя обработчик запросов клиента на получение информации о корзине и запросов на её редактирование.

Класс CartPageController содержит следующие поля:

* REDIRECT\_TO\_CART\_PAGE: String – строка, указывающая URL для перехода на страницу с информацией о корзине;
* phoneServiceImpl: PhoneService – экземпляр класса PhoneService из блока сервис, отвечающий за работу с мобильными устройствами;
* cartService: CartService – экземпляр класса CartService из блока сервис, отвечающий за управление корзиной покупателей;
* httpSession: HttpSession – экземпляр класса HttpSession, который предоставляет инструменты для работы с HTTP-сессией;
* phoneArrayDtoValidator: PhoneArrayDtoValidator – экземпляр класса PhoneArrayDtoValidator из блока валидации, отвечающий за валидацию запросов по изменению корзины.

Класс CartPageController содержит следующие методы:

* getCart(boolean, boolean, boolean, boolean, List<Long>, Model) : String – принимает запрос от пользователя для отображения информации о корзине;
* updateCart(PhoneArrayDto, Model, BindingResult): String – принимает запрос от пользователя для обновления информации в корзине;
* deleteFromCart(Long, Model) : String – принимает запрос от пользователя для удаления продукта из корзины;
* prepareModelForEmptyCart(Cart,Model) : String – служебный метод, для создания ответа, при действиях с пустой корзиной;
* validationFailed(Cart, BindingResult, Model) : String – служебный метод, для создания ответа, при ошибках при валидации запроса.

## 3.1.4 Класс OrderOverviewPageControllerAdmin

Класс OrderOverviewPageControllerAdmin представляет из

себя обработчик запросов администратора на получение информации о заказах и запросов на их редактирование.

Класс OrderOverviewPageControllerAdmin содержит поле:

* orderServiceImpl: OrderService – экземпляр класса OrderServiceImpl из блока сервис, отвечающий за работу с заказами клиентов.

Класс OrderOverviewPageController содержит следующие методы:

* getOrderOverview(Long, Model) : String – принимает запрос от пользователя для отображения информации о заказе;
* changeOrderStatus(Long, OrderStatus): String – принимает запрос от пользователя для обновления статуса заказа;
* handleOutOfStock() : String – обрабатывает возникшие в классе исключения, и отправляет ответ на перенаправление пользователя на нужную страницу.

**3.1.5 Класс AddDeviceController**

Класс AddDeviceController представляет из себя обработчик запросов администратора на отображение страницы для добавления информации о устройствах и обрабатывает добавление устройства.

Класс AddDeviceController содержит поля:

* phoneServiceImpl: PhoneService – экземпляр класса

PhoneServiceImpl из блока сервис, отвечающий за работу с устройствами;

­– phoneAddDtoValidator: Validator – экземпляр класса PhoneAddDtoValidator, отвечающий за валидацию запроса по добавлению устройства;

* messageSourse: MessageSourse – экземпляр класса

MessageSourse отвечающий за работу с файлами свойств.

Класс AddDeviceController содержит следующие методы:

* showAddDevicePage(Model) : String – принимает запрос от администратора для отображения формы для добавления устройства;
* addDeviceToDataBase(PhoneAddDto, BindingResult, Model): String – обрабатывает запрос администратора на добавление устройства в базу данных.

**3.1.6 Класс** **UpdateDeleteDeviceController**

Класс UpdateDeleteDeviceController представляет из себя обработчик запросов администратора на отображение страницы для удалении и обновлении информации о устройствах и обрабатывает удаление устройства.

Класс UpdateDeleteDeviceController содержит поля:

* phoneServiceImpl: PhoneService – экземпляр класса

PhoneServiceImpl из блока сервис, отвечающий за работу с устройствами;

­– QUANTITY\_ON\_PAGE: Long – цифровое значение ограничивающее количество устройств на странице.

Класс UpdateDeleteDeviceController содержит следующие методы:

* showDeleteUpdatePage(String, Long, Model) : String – принимает запрос от администратора для отображения формы для удаления или обновления устройства;
* delete(Long): String – обрабатывает запрос администратора на удаление устройства из базы данных.

**3.1.7 Класс** **UpdateDeviceController**

Класс UpdateDeviceController представляет из себя обработчик запросов администратора на отображение страницы для обновления информации о устройствах и обрабатывает обновление устройства.

Класс UpdateDeleteDeviceController содержит поля:

* phoneServiceImpl: PhoneService – экземпляр класса

PhoneServiceImpl из блока сервис, отвечающий за работу с устройствами;

­– phoneAddDtoValidator: Validator – экземпляр класса PhoneAddDtoValidator, отвечающий за валидацию запроса по обновлению устройства;

– jdbcStockDao: StockDao – экземпляр класса JdbcStockDao, отвечающий за управление количеством устройст в базе данных;

* messageSourse: MessageSourse – экземпляр класса

MessageSourse отвечающий за работу с файлами свойств.

Класс UpdateDeleteDeviceController содержит следующие методы:

* showUpdatePage(Model, Long) : String – принимает запрос от администратора для отображения формы для обновления устройства;
* updateDeviceInDb(PhoneAddDto, BindingResult, Model, Long, RedirectAttributes): String – обрабатывает запрос администратора на обновление информации об устройстве в базе данных;
* handleError(): String – обрабатывает возникающие ошибки.

**3.1.8 Класс AdminMainPageController**

Класс AdminMainPageController представляет из себя обработчик запросов администратора на отображение главной страницы консоли администратора.

Класс AdminMainPageController содержит следующие методы:

* showMainAdminPage(Model, Authetication) : String – принимает запрос от администратора для отображения главной страницы.

**3.1.9 Класс LoginPageController**

Класс LoginPageController представляет из себя обработчик запросов пользователей на отображение страницы для авторизаци и обслуживает авторизацию пользователей.

Класс LoginPageController содержит следующее поле:

* messageSourse: MessageSourse – экземпляр класса

MessageSourse отвечающий за работу с файлами свойств.

Класс LoginPageController содержит следующие методы:

* getloginPage(String, Authentication) : String – принимает запрос от администратора для отображения страницы авторизации;
* login(Model) : String – обрабатывает запрос по авторизации.
  + 1. **Класс OrdersPageController**

Класс OrdersPageController представляет из себя обработчик запросов администратора для отображения страницы показа текущих заказов.

Класс OrdersPageController содержит следующее поле:

* orderServiceImpl: OrderService – экземпляр класса

OrderServiceImpl отвечающий за работу с заказами пользователей.

Класс OrdersPageController содержит следующие методы:

* getOrders(String, Authentication) : String – принимает запрос от администратора для отображения страницы с текущими заказами пользователей.
  + 1. **Класс UserManagementController**

Класс UserManagementController представляет из себя обработчик запросов администратора для отображения страницы со списком пользователей, обрабатывает запрос по обновлению их статуса.

Класс UserManagementController содержит следующие поля:

* userServiceImpl: UserService – экземпляр класса

UserServiceImpl отвечающий за работу с пользователями приложения;

– LIMIT\_ON\_PAGE: int – цифровое значение ограничивающее количество пользователей на странице.

Класс UserManagementController содержит следующие методы:

* showUserManagementPage(Model, Integer) : String – принимает запрос от администратора для отображения страницы с списком пользователей;
* updateUserStatus(UserDto): ResponseEntity<?> – обрабатывает запрос администратора на обновление статуса пользователя.

**3.1.12 Класс AboutUsPageController**

Класс AboutUsPageController представляет из себя обработчик запросов администратора для отображения страницы о магазине.

Класс AboutUsPageController содержит следующее поле:

* userServiceImpl: UserService – экземпляр класса

UserServiceImpl отвечающий за работу с пользователями.

Класс AboutUsPageController содержит следующий метод:

* showAboutUsPage(Model) : String – принимает запрос от пользователя для отображения страницы с информацией об продавце.

**3.1.13 Класс ContactsPageController**

Класс ContactsPageController представляет из себя обработчик запросов администратора для отображения страницы с информацией (телефон, почта, адрес) магазина.

Класс ContactsPageController содержит следующий метод:

* getContactsPage() : String – принимает запрос от пользователя для отображения страницы с контактными данными магазина.

**3.1.14 Класс HotPricesPageController**

Класс HotPricesPageController представляет из себя обработчик запросов пользователя для отображения страницы с списком устройств, с наиболее большим процентом скидки.

Класс HotPricesPageController содержит следующие поля:

* phoneServiceImpl: PhoneService – экземпляр класса

PhoneServiceImpl отвечающий за работу с устройствами;

– AMOUNT\_TO\_GET: int – число устройств, которые необходимо вывести.

Класс HotPricesPageController содержит следующие методы:

* getHotPricesPage(Model) : String – принимает запрос от пользователя для отображения страницы с выгодными предложениями.

**3.1.15 Класс OrderOverviewPageController**

Класс OrderOverviewPageController представляет из себя обработчик запросов пользователя для отображения страницы с обзором сделанного заказа.

Класс OrderOverviewPageController содержит следующие поля:

* orderServiceImpl: OrderService – экземпляр класса

OrderServiceImpl отвечающий за работу с заказами пользователей.

Класс OrderOverviewPageController содержит следующие методы:

* getOrderOverivew(Long, Model) : String – принимает запрос от пользователя для отображения страницы с определенным заказом;
* handleError() : String – обрабатывает возникающие исключения.

**3.1.16 Класс OrderPageController**

Класс OrderPageController представляет из себя обработчик запросов пользователя для отображения страницы для создания заказа и обработчик запроса для создания заказа.

Класс OrderPageController содержит следующие поля:

* orderServiceImpl: OrderService – экземпляр класса

OrderServiceImpl отвечающий за работу с заказами пользователей;

* cartService: CartService – экземпляр класса

CartServiceImpl отвечающий за работу с корзиной;

* httpSession: HttpSession – экземпляр класса

HttpSession предназначенного для работы с HTTP-сессией;

* orderDataDtoValidator: OrderDataDtoValidator –

экземпляр класса OrderDataDtoValidator отвечающий за валидацию запроса пользователя;

* messageSourse: MessageSourse – экземпляр класса

MessageSourse отвечающий за работу с файлами свойств.

Класс OrderPageController содержит следующие методы:

* getOrder(Long, Model) : String – принимает запрос от пользователя для отображения страницы для создания заказа;
* RedirectAttributes) : String – обрабатывает запрос для создания заказа;
* prepareModelForValidationErrors(BindingResult,

RedirectAttributes) : String – служебный метод для создания ответа

с ошибками после валидации;

* prepareModelForValidationErrors(RedirectAttributes) : String – служебный метод для создания ответа с ошибкой из-за пустой корзины;
* prepareModelForValidationErrors(BindingResult, RedirectAttributes) : String – служебный метод для переведения корзины в заказ;
* handleError() : String – обрабатывает возникающие исключения.

**3.1.17 Класс** **ProductDetailsPageController**

Класс ProductDetailsPageController представляет из себя обработчик запросов пользователя для отображения страницы с подробной информацией об устройстве.

Класс ProductDetailsPageController содержит следующие поля:

* phoneServiceImpl: PhoneService – экземпляр класса

PhoneServiceImpl отвечающий за работу с устройствами;

* cartService: CartService – экземпляр класса

CartServiceImpl отвечающий за работу с корзиной;

* httpSession: HttpSession – экземпляр класса

HttpSession предназначенного для работы с HTTP-сессией.

Класс ProductDetailsPageController содержит следующие методы:

* showProductDetailsInfo(Long, Model) : String – принимает запрос от пользователя для отображения страницы с информацией об устройстве.

**3.1.18 Класс QuickOrderController**

Класс QuickOrderController представляет из себя обработчик запросов пользователя для создания быстрого заказа в системе и отображения соответствующей страницы.

Класс QuickOrderController содержит следующие поля:

* phoneServiceImpl: PhoneService – экземпляр класса

PhoneServiceImpl отвечающий за работу с устройствами;

* cartService: CartService – экземпляр класса

CartServiceImpl отвечающий за работу с корзиной;

* httpSession: HttpSession – экземпляр класса

HttpSession предназначенного для работы с HTTP-сессией;

* phoneList: List<Phone> – список добавленных телефонов;

Класс QuickOrderController содержит следующие методы:

* getQuickOrderPage(Model) : String – принимает запрос от пользователя для отображения страницы для быстрого заказа;
* makeQuickOrder(QuickOrderElemetsDto, BindingResult, RedirectAttributes, Model) : String – принимает запрос от пользователя для быстрого добавления в карту устройств;
* addToCart(QuickOrderElemetsDto, BindingResult, Cart, int) : void – служебный метод для поочередного добавления товаров в корзину.

**3.1.19 Класс RegisterPageController**

Класс RegisterPageController представляет из себя обработчик запросов пользователя для страницы с регистрацией, обрабатывает запрос на регистрацию.

Класс RegisterPageController содержит следующие поля:

* userServiceImpl: UserService – экземпляр класса

UserServiceImpl отвечающий за работу с пользователями магазина.

Класс RegisterPageController содержит следующие методы:

* showRegisterPage() : String – принимает запрос от пользователя для отображения страницы для регистрации;
* registerUser(String, String, Model) : String – принимает запрос от пользователя для регистрации в системе.

## 3.1.20 Интерфейс PhoneshopPages

Интерфейс PhoneshopPages представляет своеобразное хранилище имён веб-страниц, на которые перенаправляются запросы.

Интерфейс PhoneshopPages содержит имена веб страниц (HotPricesPage, LoginPage, CartPage, OrderOverviewPage, OrderPage, ProductDetailsPage, ProductListPage, QuickOrderPage, OrdersPage,OrderOverviewPageAdmin, AdminMainPage, AddDevicePage, UpdateDeletePage, UserManagementPage, AboutUsPage, ContactsPage, RegisterPage).

**3.1.21 Класс OrderDataDto**

Класс OrderDataDto описывает запрос пользователя, переданный при создании заказа.

Класс OrderDataDto содержит следующие поля:

* firstName: String – имя заказчика;
* lastName: String – фамилия заказчика;
* deliveryAddress: String – адрес заказчика;
* contactPhoneNo: String – телефонный номер заказчика;
* additionalInfo: String – дополнительная информация.

Класс OrderDataDto содержит следующие методы:

* getFirstName (): String – получение значения поля firstName;
* getLastName (): String – получение значения поля lastName;
* getDeliveryAddress(): String – получение значения поля deliveryAddress;
* getContactPhoneNumber(): String – получение значения поля contactPhoneNo;
* getAdditionalInfo(): String – получение значения поля additionalInfo;
* setFirstName(String): void – установка значения в поле firstName;
* setLastName (String): void – установка значения в поле lastName;
* setDeliveryAddress(String): void– установка значения в поле deliveryAddress;
* setContactPhoneNumber(String): void – установка значения в поле contactPhoneNo;
* setAdditionalInfo(String): void – установка значения в поле additionalInfo.

**3.1.22 Класс PhoneAddDto**

Класс PhoneAddDto описывает запрос администратора при добавлении устройства.

Класс PhoneAddDto содержит следующие поля:

* brand: String – брэнд-производитель устройства;
* model: String – модель устройства;
* price: BigDecimal – цифровое значение цены устройства;
* discountPercent: Integer – число, описывающее процент скидки;
* displaySizeInches: BigDecimal – размер экрана устройства в пикселях;
* weightGr: Integer – вес устройства в граммах;
* lengthMm: BigDecimal – длина устройства;
* widthMm: BigDecimal – ширина устройства;
* heightMm: BigDecimal – высота устройства;
* announced: Date – дата презентации устройства;
* deviceType: String – тип устройства (телефон, планшет);
* colors: String – цветовая гамма телефонов;
* displayResolution: String – разрешение экрана устройства;
* pixelDensity: Integer – плотность пикселей устройства;
* displayTechnology: String – технология экрана устройства;
* backCameraMegapixels: BigDecimal – мегапиксели задней камеры;
* frontCameraMegapixels: BigDecimal – мегапиксели передней камеры;
* batteryCapacityMah: Integer – мощность аккумуляторной батареи;
* talkTimeHours: BigDecimal – время разговора;
* standByTimeHours: BigDecimal – время работы в режиме ожидания;
* bluetooth: String – технология bluetooth;
* imageUrl: String – ссылка на изображение;
* description: String – описание устройства.

Класс PhoneAddDto содержит методы для получения и установки значений в объектах класса (геттеры и сеттеры). Также в классе описаны методы equals и hashCode, используемые при сравнении устройств.

**3.1.23 Класс PhoneArrayDto**

Класс PhoneArrayDto описывает запрос пользователя, переданный при попытке обновления корзины.

Класс PhoneArrayDto содержит следующие поля:

* phoneId: String[] – набор идентификаторов устройств;
* quantity: String[] – набор количеств устройств.

Класс PhoneArrayDto содержит следующие методы:

* getPhoneId(): String[] – получение значения поля phoneId;
* getQuantity (): String – получение значения поля quantity;
* setPhoneId(String[]): void – установка значения в поле phoneId;
* setQuantity(String[]): void – установка значения в поле quantity.

**3.1.24 Класс QuickOrderElement**

Класс QuickOrderElement описывает один из запросов пользователя, переданный при быстром заказе.

Класс QuickOrderElement содержит следующие поля:

* model: String – строка, описывающая модель устройства;
* quantity: Long – количество элементов.

Класс QuickOrderElement содержит следующие методы:

* getModel(): String – получение значения поля model;
* getQuantity (): Long – получение значения поля quantity;
* setModel(String): void – установка значения в поле model;
* setQuantity(Long): void – установка значения в поле quantity.

**3.1.25 Класс** **QuickOrderElementsDto**

Класс QuickOrderElementsDto описывает запрос пользователя, переданный при быстром заказе.

Класс QuickOrderElementsDto содержит следующие поля:

* quickOrderElements: List<QuickOrderElements> – набор экземпляров класса QuickOrderElement, из которого формируется весь быстрый заказ.

Класс QuickOrderElementsDto содержит следующие методы:

* getQuickOrderElements(): List<QuickOrderElement> – получение значения поля quickOrderElements;
* setQuickOrderElements(List<QuickOrderElement>): void – установка значения в поле quickOrderElements;
* toString(): String – отображение всех полей класса в символьном представлении.

**3.1.26 Класс PhoneDto**

Класс PhoneDto описывает запрос пользователя, переданный при попытке попытке добавления в корзину.

Класс PhoneDto содержит следующие поля:

* id: Long– идентификатор устройства;
* quantity: Long – количество добавляемых в корзину устройств.

Класс PhoneDto содержит следующие методы:

* getId(): Long – получение значения поля id;
* getQuantity(): Long – получение значения поля quantity;
* setId(Long): void – установка значения в поле id;
* setQuantity(Long): void – установка значения в поле quantity.

**3.1.27 Класс** **UserDto**

Класс UserDto описывает запрос администратора переданный при блокироке/разблокировке пользователя.

Класс UserDto содержит следующие поля:

* userName: String – строка, описывающая имя пользователя;
* status: Integer – статус аккаунта пользователя (блокирован или нет).

Класс UserDto содержит следующие методы:

* getUserName(): String – получение значения поля userName;
* getStatus (): Integer – получение значения поля status;
* setUserName(String): void – установка значения в поле userName;
* setStatus(Integer): void – установка значения в поле status.

**3.1.28 Класс UserRegisterDto**

Класс UserRegisterDto описывает запрос администратора переданный при регистрации пользователя.

Класс UserRegisterDto содержит следующие поля:

* userName: String – строка, описывающая имя пользователя;
* password: String – строка, описывающая пароль от учётной записи.

Класс UserRegisterDto содержит следующие методы:

* getUserName(): String – получение значения поля userName;
* getPassword (): String – получение значения поля password;
* setUserName(String): void – установка значения в поле userName;
* setPassword(String): void – установка значения в поле password;
* toString(): String – отображение всех полей класса в символьном представлении.

**3.2 Блок моделей**

**3.2.1 Класс Phone**

Класс Phone представляет описание сущности мобильного устройства

Класс Phone содержит следующие поля:

* id: Long – уникальный идентификатор устройства;
* brand: String – брэнд-производитель устройства;
* model: String – модель устройства;
* price: BigDecimal – цифровое значение цены устройства;
* discountPercent: Integer – число, описывающее процент скидки;
* actualPrice: BigDecimal – цена устройства с учётом скидки;
* displaySizeInches: BigDecimal – размер экрана устройства в пикселях;
* weightGr: Integer – вес устройства в граммах;
* lengthMm: BigDecimal – длина устройства;
* widthMm: BigDecimal – ширина устройства;
* heightMm: BigDecimal – высота устройства;
* announced: Date – дата презентации устройства;
* deviceType: String – тип устройства (телефон, планшет);
* colors: Set<Colors> – цветовая гамма телефонов;
* displayResolution: String – разрешение экрана устройства;
* pixelDensity: Integer – плотность пикселей устройства;
* displayTechnology: String – технология экрана устройства;
* backCameraMegapixels: BigDecimal – мегапиксели задней камеры;
* frontCameraMegapixels: BigDecimal – мегапиксели передней камеры;
* internalStorageGb: BigDecimal – объём внутренней памяти;
* batteryCapacityMah: Integer – мощность аккумуляторной батареи;
* talkTimeHours: BigDecimal – время разговора;
* standByTimeHours: BigDecimal – время работы в режиме ожидания;
* bluetooth: String – технология bluetooth;
* positioning: String – позиционирование устройства;
* imageUrl: String – ссылка на изображение;
* description: String – описание устройства.

Класс Phone содержит методы для получения и установки значений в объектах класса (геттеры и сеттеры). Также в классе описаны методы equals и hashCode, используемые при сравнении устройств.

**3.2.2 Класс Cart**

Класс Cart описывает один виртуальную корзину из которой формируется заказ.

Класс Cart содержит следующие поля:

* cartItems: List<CartItems> – список элементов в корзине;
* totalQuantity: Long – общее количество элементов в корзине;
* totalCost: BigDecimal – обозначает общую цену корзины.

Класс Cart содержит следующие методы:

* CartItem()– конструктор для инициализации полей экземпляра класса;
* getCartItems(): List<CartItem> – получение значения поля cartItems;
* getTotalQuantity(): Long – получение значения поля totalQuantity;
* getTotalCost (): BigDecimal – получение значения поля totalCost;
* setCartItems(List<CartItems>): void – установка значения в поле cartItems;
* setTotalQantity(Long): void – установка значения в поле totalQuantity;
* setTotalCost(BigDecimal): void – установка значения в поле totalCost;
* toString(): String – отображение всех полей класса в символьном представлении.

**3.2.3 Класс Color**

Класс Color описывает цвета, которые могут содержаться в

устройстве.

Класс Color содержит следующие поля:

* id: Long – уникальный идентификатор цвета;
* code: String – строка, обозначающая имя цвета.

Класс Color содержит следующие методы:

* getId(): Long – получение значения поля id;
* getCode (): String – получение значения поля code;
* setCode(String): void – установка значения в поле code;
* setId(Long): void – установка значения в поле id.

**3.2.4 Класс Employee**

Класс Employee описывает сотрудников компании (имя, фамилия, должность, тип).

Класс Employee содержит следующие поля:

* id: Long – уникальный идентификатор сотрудника;
* firstName: String – имя сотрудника;
* lastName: String – фамилия сотрудника;
* position: String – должность сотрудника;
* employeeType: EmployeeType – перечисление, используемое для определения типа сотрудника.

Класс Employee содержит следующие методы:

* getId(): Long – получение значения поля id;
* getFirstName (): String – получение значения поля firstName;
* getLastName (): String – получение значения поля lastName;
* getPosition (): String – получение значения поля position;
* getEmployeeType (): EmployeeType – получение значения поля employeeType;
* setFirstName(String): void – установка значения в поле firstName;
* setId(Long): void – установка значения в поле id;
* setLastName(String): void – установка значения в поле lastName;
* setPosition(String): void – установка значения в поле position;
* toString(): String – вывод полей объекта классов в строковом представлении;
* setEmployeeType(EmployeeType): void – установка значения в поле employeeType.

**3.2.5 Класс Order**

Класс Order описывает заказы, создаваемые пользователями.

Класс Order содержит следующие поля:

* id: Long – уникальный идентификатор сотрудника;
* firstName: String – имя заказчика;
* lastName: String – фамилия заказчика;
* deliveryAddress: String – адрес заказчика;
* contactPhoneNo: String – телефонный номер заказчика;
* additionalInfo: String – дополнительная информация;
* date: Timestamp – дата заказа;
* orderStatus: OrderStatus – перечисление описывающее статус заказа;
* orderItems: List<OrderItem> – список элементов заказа;
* subtotal: BigDecimal – общая стоимость заказа;
* deliveryPrice: BigDecimal – стоимость доставки;
* totalPrice: BigDecimal – стоимость заказа с доставкой.

Класс Order содержит следующие методы:

– Order()– конструктор для инициализации полей объекта класса;

* getId(): Long – получение значения поля id;
* getFirstName (): String – получение значения поля firstName;
* getLastName (): String – получение значения поля lastName;
* getDeliveryAddress(): String – получение значения поля deliveryAddress;
* getContactPhoneNumber(): String – получение значения поля contactPhoneNo;
* getAdditionalInfo(): String – получение значения поля additionalInfo;
* getDate(): Timestamp – получение значения поля date;
* getOrderStatus(): OrderStatus – получение значения поля status;
* getOrderItems(): List<OrderItem> – получение значения поля orderItems;
* getSubtotal(): BigDecimal – получение значения поля subtotal;
* getDeliveryPrice(): BigDecimal – получение значения поля deliveryPrice;
* getTotalPrice(): BigDecimal – получение значения поля totalPrice;
* setId(Long): void– установка значения в поле id;
* setFirstName(String): void – установка значения в поле firstName;
* setLastName (String): void – установка значения в поле lastName;
* setDeliveryAddress(String): void– установка значения в поле deliveryAddress;
* setContactPhoneNumber(String): void – установка значения в поле contactPhoneNo;
* setAdditionalInfo(String): void – установка значения в поле additionalInfo;
* setDate(Timestamp): void – установка значения в поле date;
* setOrderStatus(OrderStatus): void – установка значения в поле status;
* setOrderItems(List<OrderItem>):– установка значения в поле orderItems;
* setSubtotal(BigDecimal): void– установка значения в поле subtotal;
* setDeliveryPrice(BigDecimal): void– установка значения в поле deliveryPrice;
* setTotalPrice(BigDecimal): void – установка значения в поле totalPrice.

**3.2.6 Класс OrderItem**

Класс OrderItem описывает один из элементов, находящийся в пользовательском заказе.

Класс OrderItem содержит следующие поля:

* id: Long – уникальный идентификатор;
* phone: Phone – экземпляр класса Phone из блока моделей, представляющий устройство;
* quantity: Long – обозначает количество устройств в данном элементе;
* order: Order – экземпляр класса Order, ссылающийся на заказ.

Класс OrderItem содержит следующие методы:

* OrderItem (Phone, Long, Order, Long)– конструктор для инициализации полей экземпляра класса;
* OrderItem (CartItem, Order)– конструктор для инициализации полей экземпляра класса;
* getId(): Long – получение значения поля id;
* getPhone(): Phone – получение значения поля phone;
* getQuantity(): Long – получение значения поля quantity;
* getOrder(): Order – получение значения поля order;
* setId(Long): void – установка значения в поле id;
* setPhone(Phone): void – установка значения в поле phone;
* setQuantity(Long): void – установка значения в поле quantity;
* setOrder(Order): void – установка значения в поле order.

**3.2.7 Класс CartItem**

Класс CartItem описывает один из элементов, находящийся в пользовательской корзине.

Класс CartItem содержит следующие поля:

* phone: Phone – экземпляр класса Phone из блока моделей, представляющий устройство;
* quantity: Long – обозначает количество телефонов в данном элементе;
* price: BigDecimal – обозначает общую цену элемента.

Класс CartItem содержит следующие методы:

* CartItem(Phone, Long, BigDecimal)– конструктор для инициализации полей экземпляра класса;
* getPhone(): Phone – получение значения поля phone;
* getQuantity(): Long – получение значения поля quantity;
* getPrice (): BigDecimal – получение значения поля price;
* setPhone(Phone): void – установка значения в поле phone;
* setQuantity(Long): void – установка значения в поле quantity;
* setPrice(BigDecimal): void – установка значения в поле price;
* toString(): String – вывод значений объекта класса в строковом представлении.

**3.2.8 Класс ParamsForSearch**

Класс ParamsForSearch позволяет обобщить элементы поиска, сортировки, определения страницы для вывода и оптимизировать код программы.

Класс ParamsForSearch содержит следующие поля:

* search: String – строка с поисковым запросом;
* sortField: String – строка с названием поля для сортировки;
* order: String – строка для обозначения порядка сортировки;
* offset: int – число, обозначающее смещение от первого элемента;
* limit: int – число, обозначающее максимальное количество элементов для поиска.

Класс ParamsForSearch содержит следующие методы:

* ParamsForSearch(String, String, String, int, int)– конструктор для инициализации полей экземпляра класса;
* getSearch(): String – получение значения поля search;
* getSortField(): String – получение значения поля sortField;
* getOrder(): String – получение значения поля order;
* getLimit(): int – получение значения поля limit;
* getOffset(): String – получение значения поля offset;
* setSearch(String): void – установка значения в поле search;
* setSortField(String): void – установка значения в поле sortField;
* setOrder(String): void – установка значения в поле order;
* setLimit(int): void – установка значения в поле limit;
* setOffset(String): void – установка значения в поле offset.

**3.2.9 Класс Stock**

Класс Stock описывает наличие устройств в магазине и зарезервированное в корзинах количество устройств.

Класс Stock содержит следующие поля:

* phone: Phone – строка с поисковым запросом;
* stock: Integer – строка с названием поля для сортировки;
* reserved: Integer – строка для обозначения порядка сортировки.

Класс Stock содержит следующие методы:

* getPhone(): Phone – получение значения поля phone;
* getStock(): Integer – получение значения поля stock;
* getReserved(): Integer – получение значения поля reserved;
* setPhone(Phone): void – установка значения в поле phone;
* setStock(Integer): void – установка значения в поле stock;
* setReserved(Integer): void – установка значения в поле reserved.

**3.2.10 Класс User**

Класс User описывает пользователей сайта, данные для авторизации,

права пользователя, статус аккаунта.

Класс User содержит следующие поля:

* userName: String – строка хранящая в себе имя пользователя;
* password: String – строка хранящая в себе закодированный пароль;
* isAccountNonLocked: int – цифровое значение описывающее статус аккаунта(блокирован или нет);
* role: String – строка содержащая информацию о правах пользователя;

Класс User содержит следующие методы:

* getUserName(): String – получение значения поля userName;
* getPassword(): String – получение значения поля password;
* getIsAccountNonLocked(): int – получение значения поля isAccountNonLocked;
* getRole(): String – получение значения поля role;
* setUserName(String): void – установка значения в поле userName;
* setPassword(String):void – установка значения в поле password;
* setIsAccountNonLocked(int): void – установка значения в поле isAccountNonLocked;
* setRole(String): void – установка значения в поле role;
* toString(): String – отображение всех полей класса в символьном представлении.

**3.2.11 Перечисление OrderStatus**

Перечисление OrderStatus отображает четыре статуса обработки заказа:

* NEW – новый заказ;
* REJECTED – заказ отменен;
* IN­­DELIVERY – заказ доставляется;
* DELIVERED – заказ доставлен.

**3.2.12 Перечисление SortField**

Перечисление SortField обозначает поля, доступные для сортировки:

* BRAND – сортировка по брэнду;
* MODEL – сортировка по модели;
* DISPLAYSIZEINCHES – сортировка по размеру заказа;
* PRICE – сортировка по цене.

**3.2.13 Перечисление SortOrder**

Перечисление SortOrder обозначает поля, доступные для сортировки:

* ASC – сортировка по возрастанию;
* DESC – сортировка по убыванию.

**3.2.14 Перечисление** **EmployeeType**

Перечисление EmployeeType отображает 3 вида сотрудников (в частности это используется для выбора изображения):

* MALE – сотрудник-мужчина;
* FEMALE – сотрудник-женщина;
* GEEK\_MALE – сотрудник-мужчина технический специалист.

**3.3 Блок работы с СУБД**

**3.3.1 Интерфейс ColorDao**

Интерфейс ColorDao объявляет набор функций, которые потребуются для взаимодействия с таблицей colors из базы данных, интерфейс реализуется классом JdbcColorDao.

Интерфейс ColorDao содержит методы:

* save(Color) : void – выполняет sql-запрос для сохранения цвета в базе данных;
* get(Long) : Optional<Color> – выполняет sql-запрос для получения информации о цвете из базы данных.

**3.3.2 Интерфейс OrderDao**

Интерфейс OrderDao объявляет набор функций, которые потребуются для взаимодействия с таблицей orders из базы данных, интерфейс реализуется классом JdbcOrderDao.

Интерфейс OrderDao содержит методы:

* save(Order) : void – выполняет sql-запрос для сохранения информации об заказе в базе данных;
* get(Long) : Optional<Order> – выполняет sql-запрос для получения информации об заказе из базы данных;
* getOrders(int, int) : List<Order> – выполняет sql-запрос для получения некоторого количества заказов из базы данных;
* updateStatus(OrderStatus, Long) : void – выполняет sql-запрос для обновления статуса заказа в базе данных.

**3.3.3 Интерфейс** **PhoneDao**

Интерфейс PhoneDao объявляет набор функций, которые потребуются для взаимодействия с таблицей phones из базы данных, интерфейс реализуется классом JdbcPhoneDao.

Интерфейс PhoneDao содержит методы:

* get(Long) : Optional<Phone> – выполняет sql-запрос для получения устройства из базы данных по уникальному идентификатору;
* get(String) : Optional<Phone> – выполняет sql-запрос для сохранения цвета в базе данных;
* getPhoneAndColors(String) : Optional<Phone> – объединяет результат запросов по получению телефонов и цветов к ним;
* save(Phone) : void – сохраняет устройство в базе данных;
* findAll(int, int) : List<Phone> – поиск телефонов со смещением;
* findAll(ParamsForSearch) : List<Phone> – поиск телефонов с параметрами поиска;
* count(ParamsForSearch) : Long – подсчёт количества телефонов;
* findMaxDiscountPercentPhones(int, int) : List<Phone> – поиск телефонов со скидкой, отсортированной по

убыванию;

* deletePhone(Long) : void – удаление устройства из базы данных;
* updateDevice(Phone) : void – обновление информации об устройстве в базе данных.

**3.3.4 Интерфейс StockDao**

Интерфейс StockDao объявляет набор функций, которые потребуются для взаимодействия с таблицей stock из базы данных, интерфейс реализуется классом JdbcStockDao.

Интерфейс StockDao содержит методы:

* update(Long, Long, Long) : void – выполняет sql-запрос для сохранения наличных устройств в базе данных;
* get(Long) : Optional<Stock> – выполняет sql-запрос для получения информации о наличии устройств из базы данных.

**3.3.5 Интерфейс UserDao**

Интерфейс UserDao объявляет набор функций, которые потребуются для взаимодействия с таблицей users из базы данных, интерфейс реализуется классом JdbcUserDao.

Интерфейс UserDao содержит методы:

* getUserByUsername(String) : Optional<User> – получение пользователя из базы данных по его имени;
* getUsersExceptAdmin(int, int) : List<User> – получение списка пользователей из базы данных кроме администратора;
* updateUserStatus(int, String) : void – обновление статуса пользователя в базе данных;
* addNewUserToDb(User) : void – добавление нового пользователя в базу данных;
* getMainEmployees(): void –получение списка сотрудников.

**3.3.6 Класс JdbcColorDao**

Класс JdbcColorDao представляет из себя набор утилит для осуществления SQL-запросов для получения информации о цветах телефонов.

Класс JdbcColorDao содержит следующие поля:

* jdbcTemplate: JdbcTemplate – экземпляр класса JdbcTemplate служащего для отправки SQL-запросов и упаковки ответов в объекты;
* SQL\_GET\_COLOR: String – SQL-запрос для получения цвета по идентификатору;
* SQL\_UPDATE\_COLOR: String – SQL-запрос для сохранения цвета в базе данных.

Класс JdbcColorDao содержит методы:

* save(Color) : void – выполняет sql-запрос для сохранения цвета в базе данных;
* get(Long) : Optional<Color> – выполняет sql-запрос для получения информации о цвете из базы данных.

**3.3.7 Класс JdbcPhoneDao**

Класс JdbcPhoneDao представляет из себя набор утилит для осуществления SQL-запросов для получения информации о телефонах.

Класс JdbcPhoneDao содержит следующие поля:

­ ­– jdbcTemplate: JdbcTemplate – экземпляр класса JdbcTemplate служащего для отправки SQL-запросов и упаковки ответов

в объекты;

* namedParameterjdbcTemplate: JdbcTemplate – экземпляр класса NamedParameterJdbcTemplate служащего для отправки SQL-запросов и упаковки ответов в объекты;
* phoneResultSetExtractor: PhoneResultSetExtractor – экземпляр класса PhoneResultSetExtractor служащий для преобразования ответов в java-объекты;
* jdbcColorDao: JdbcColorDao – экземпляр класса JdbcColorDao служащий для получения информации о цветах телефонов;
* SQL\_INSERT\_PHONE: String – SQL-запрос для сохранения устройства в базе данных;
* SQL\_GET\_PHONE: String – SQL-запрос для получения устройства из базы данных;
* SQL\_GET\_ALL\_PHONES: String – SQL-запрос для получения устройств из базы данных;
* SQL\_GET\_MAX\_DISCOUNT\_PHONES: String – SQL-запрос для получения устройств со скидкой из базы данных;
* SQL\_GET\_COUNT\_PHONES: String – SQL-запрос для получения количества устройств из базы данных;
* SQL\_WHERE\_SEARCH: String – дополнительные условия для запросов описанных выше;
* SQL\_SELECT\_COLOR\_IDS: String – SQL-запрос для получения цветов телефона по его идентификатору;
* SQL\_GET\_PHONE\_BY\_MODEL: String – SQL-запрос для получения устройства по названию модели;
* DELETE\_PHONE\_SQL: String – SQL-запрос для удаления устройства из базы данных;
* SQL\_UPDATE\_PHONE: String – SQL-запрос для обновления информации об устройстве в базе данных;

Класс JdbcPhoneDao содержит методы:

* get(Long) : Optional<Phone> – выполняет sql-запрос для получения устройства из базы данных по уникальному идентификатору;
* get(String) : Optional<Phone> – выполняет sql-запрос для сохранения цвета в базе данных;
* getPhoneAndColors(String) : Optional<Phone> – объединяет результат запросов по получению телефонов и цветов к ним;
* save(Phone) : void – сохраняет устройство в базе данных;
* findAll(int, int) : List<Phone> – поиск телефонов со смещением;
* findAll(ParamsForSearch) : List<Phone> – поиск телефонов с параметрами поиска;
* count(ParamsForSearch) : Long – подсчёт количества телефонов;
* findMaxDiscountPercentPhones(int, int) : List<Phone> – поиск телефонов со скидкой, отсортированной по убыванию;
* deletePhone(Long) : void – удаление устройства из базы данных;
* updateDevice(Phone) : void – обновление информации об устройстве в базе данных;
* populateNamedParams(Phone): MapSqlParameterSource – служебный метод для задания именных параметров для SQL-запросов (save, update).

**3.3.8 Класс JdbcStockDao**

Класс JdbcStockDao представляет из себя набор утилит для осуществления SQL-запросов для получения информации о наличии устройств в базе данных.

Класс JdbcStockDao содержит следующие поля:

* jdbcTemplate: JdbcTemplate – экземпляр класса JdbcTemplate служащего для отправки SQL-запросов и упаковки ответов в объекты;
* SQL\_GET\_STOCK: String – SQL-запрос для получения цвета по идентификатору устройства;
* SQL\_UPDATE: String – SQL-запрос для обновления информации об наличии в базе данных;
* SQL\_INSERT: String – SQL-запрос для создания информации об наличии в базе данных.

Класс JdbcStockDao содержит методы:

* update(Long, Long, Long) : void – выполняет sql-запрос для сохранения наличных устройств в базе данных;
* get(Long) : Optional<Stock> – выполняет sql-запрос для получения информации о наличии устройств из базы данных.

**3.3.9 Класс JdbcOrderDao**

Класс JdbcOrderDao представляет из себя набор утилит для осуществления SQL-запросов для получения информации об заказах в базе данных.

Класс JdbcOrderDao содержит следующие поля:

– jdbcTemplate: JdbcTemplate – экземпляр класса

JdbcTemplate служащего для отправки SQL-запросов и упаковки ответов в объекты;

* namedParameterJdbcTemplate: JdbcTemplate – экземпляр класса NamedParameterJdbcTemplate служащего для отправки SQL-запросов;
* orderResultSetExtractor: OrderResultSetExtractor – экземпляр класса OrderResultSetExtractor преобразующий sql-ответы в объект класса Order;
* orderListResultSetExtractor:OrderListResultSetExtractor – экземпляр класса OrderListResultSetExtractor преобразующий sql-ответы в объект класса List<Order>;
* SELECT\_ORDER\_SQL: String – SQL-запрос для получения заказа по идентификатору;
* INSERT\_ORDER\_SQL: String – SQL-запрос для сохранения заказа в базе данных;
* SQL\_INSERT\_INTO\_ORDER\_ITEMS: String – SQL-запрос для сохранения элементов заказа в базе данных;
* SQL\_SELECT\_ORDERS: String – SQL-запрос для получения нескольких заказов;
* UPDATE\_ORDER\_STATUS\_SQL: String – SQL-запрос для обновления статуса заказа;
* SQL\_NEEDED\_ORDER: String – SQL-запрос для составления дополнительных условий поиска.

Класс JdbcOrderDao содержит методы:

* save(Order) : void – выполняет sql-запрос для сохранения информации об заказе в базе данных;
* get(Long) : Optional<Order> – выполняет sql-запрос для получения информации об заказе из базы данных;
* getOrders(int, int) : List<Order> – выполняет sql-запрос для получения некоторого количества заказов из базы данных;
* updateStatus(OrderStatus, Long) : void – выполняет sql-запрос для обновления статуса заказа в базе данных.

**3.3.10 Класс JdbcUserDao**

Класс JdbcUserDao представляет из себя набор утилит для осуществления SQL-запросов для получения информации об пользователях в базе данных.

Класс JdbcUserDao содержит следующие поля:

* jdbcTemplate: JdbcTemplate – экземпляр класса

JdbcTemplate служащего для отправки SQL-запросов и упаковки ответов

в объекты;

* resultSetExtractor: UserListResultSetExtractor – экземпляр класса UserListResultSetExtractor преобразующий sql-ответы в объект класса List<User>;
* employeeListResultSetExtractor:EmployeeListResu-ltSetExtractor экземпляр класса EmployeeListResultSetExtractor преобразующий sql-ответы в объект класса List<Employee>;
* SQL\_GET\_EMPLOYEES: String – SQL-запрос для получения списка сотрудников;
* SQL\_GET\_USER: String – SQL-запрос для пользователя из базы данных;
* SQL\_GET\_USERS\_EXCEPT\_ADMIN: String – SQL-запрос для получения списка пользователей, кроме администратора;
* SQL\_UPDATE\_USER\_STATUS: String – SQL-запрос для обновления статуса пользователя;
* SQL\_INSERT\_NEW\_USER: String – SQL-запрос для создания новой записи о пользователе в базе данных.

Класс JdbcUserDao содержит методы:

* getUserByUsername(String) : Optional<User> – получение пользователя из базы данных по его имени;
* getUsersExceptAdmin(int, int) : List<User> – получение списка пользователей из базы данных кроме администратора;
* updateUserStatus(int, String) : void – обновление статуса пользователя в базе данных;
* addNewUserToDb(User) : void – добавление нового пользователя в базу данных;
* getMainEmployees(): void –получение списка сотрудников.

**3.4 Блок сервис**

**3.4.1 Интерфейс OrderService**

Интерфейс OrderService объявляет набор функций, которые потребуются для взаимодействия с пользовательскими заказами, интерфейс реализуется классом OrderServiceImpl.

Интерфейс OrderService содержит методы:

* placeOrder(Cart, OrderDataDto, Long) : Long – выполняет логику по созданию заказа, обновления наличия устройств, вызывает методы блока работы с СУБД для сохранения информации в базе данных;
* getOrders(int, int) : List<Order> – выполняет логику для получения определенных заказов;
* getOrderById(Long) : Optional<Order> – выполняет логику для получения заказа по идентификатору;
* updateStatus(OrderStatus, Long) : void – вызывает методы из блока работы с СУБД, для обновления статуса заказа.

**3.4.2 Интерфейс** **CartService**

Интерфейс CartService объявляет набор функций, которые потребуются для взаимодействия с пользовательской корзиной, интерфейс реализуется классом HttpSessionCartService.

Интерфейс CartService содержит методы:

* getCart(HttpSession) : Cart – создаёт или получает значения атрибута корзины из сессии;
* addPhone(Long, Long, Cart) : void – добавляет продукт в корзину;
* addToCart(Long, Stock, Phone, Cart) : void – проверяет существует ли продукт в корзине и в зависимости от результата

создает новый или обновляет существующий;

* update(Map<Long,Long>, Cart) : void – обновляет информацию в корзине;
* remove(Long, Cart) : void – удаляет элемент из корзины;
* deleteCart(HttpSession) : void – удаляет корзину из сессии.

**3.4.3 Интерфейс PhoneService**

Интерфейс PhoneService объявляет набор функций, которые потребуются для взаимодействия с продаваемыми устройствами, интерфейс реализуется классом PhoneServiceImpl.

Интерфейс PhoneService содержит методы:

* getPhoneByKey(Long) : Optional<Phone> – получает устройство по уникальному идентификатору;
* getPhoneByModel(String) : Optional<Phone> – получает устройство по названию модели;
* savePhone(Phone, Long, String) : void – сохраняет телефон и необходимую информацию в базе данных;
* getPhones(int, int) : List<Phone> – получить список устройств из базы данных с определенным смещением и ограничением;
* getPhones(ParamsForSearch) : List<Phone> – получить список устройств из базы данных с условием поиска, сортировки, определенным смещением и ограничением;
* populateFields(PhoneAddDto) : Phone – преобразование DTO-объекта в модель Phone;
* deletePhone(Long) : void – удалить устройство и связанную с ним информацию из системы;

– updateDevice(Phone, Long, String[] colors, Long) :

void – обновляет информацию об устройстве в системе;

* countPhoneQuantity(ParamsForSearch) : Long – подсчёт количества устройств в системе;
* getHotPricedPhones(int) : List<Phone> – получить телефоны со скидкой.

**3.4.4 Интерфейс UserService**

Интерфейс UserService объявляет набор функций, которые потребуются для взаимодействия с пользователями системы и сотрудниками, интерфейс реализуется классом UserServiceImpl.

Класс UserService содержит методы:

* loadUserByUsername(String) : User – получить информацию о пользователе по его имени учётной записи;
* getUsersExceptAdmin(int, int) : List<User> – получить список пользователей кроме администратора;
* updateUserStatus(int, String) : void – обновляет статус пользователя в система;
* registerUser(String, String) : void – регистрирует пользователя в системе;
* getEmployees() : List<Employee> – получить список сотрудников магазина.

**3.4.5 Класс OrderServiceImpl**

Класс OrderServiceImpl представляет из себя набор утилит для осуществления бизнес-логики для работы над заказами и абстракции блока обработки запросов от блока работы с СУБД.

Класс OrderServiceImpl содержит следующие поля:

* jdbcOrderDao: OrderDao – экземпляр класса JdbcOrderDao служащего для работы с частью СУБД, связанной с заказами;
* jdbcStockDao: StockDao – экземпляр класса JdbcStockDao служащего для работы с частью СУБД, связанной с количеством устройств в

наличии.

Класс OrderServiceImpl содержит методы:

* placeOrder(Cart, OrderDataDto, Long) : Long – выполняет логику по созданию заказа, обновления наличия устройств, вызывает методы блока работы с СУБД для сохранения информации в базе данных;
* getOrders(int, int) : List<Order> – выполняет логику для получения определенных заказов;
* getOrderById(Long) : Optional<Order> – выполняет логику для получения заказа по идентификатору;
* updateStatus(OrderStatus, Long) : void – вызывает методы из блока работы с СУБД, для обновления статуса заказа;
* createOrderFromCart(Cart, OrderDataDto, OrderStatus, Long) : Order – преобразовывает корзину с покупками в заказ.

**3.4.6 Класс** **AuthProvider**

Класс AuthProvider представляет из себя набор утилит для осуществления бизнес-логики для работы с пользователями и

абстракции блока обработки запросов от блока работы с СУБД.

Класс AuthProvider содержит следующие поля:

* userDetailsService: UserService – экземпляр класса UserServiceImpl служащего для работы в СУБД с пользователями приложения.

Класс AuthProvider содержит методы:

* authenticate(Authentication) : Authentication – выполняет логику по шифрованию введенного пользователем пароля, сравнения имени пользователя и зашифрованного пароля из базы данных и в зависимости от результата предоставляет права пользователю;
* supports(aClass: Class<?>) : boolean – выполняет проверку на возможность валидации реквизитов для входа предоставленного класса.

**3.4.7 Класс HttpSessionCartService**

Класс HttpSessionCartService представляет из себя набор утилит для осуществления бизнес-логики для работы c пользовательской корзиной и абстракции блока обработки запросов от блока работы с СУБД.

Класс HttpSessionCartService содержит следующие поля:

* jdbcPhoneDao: PhoneDao – экземпляр класса JdbcPhoneDao служащего для работы с частью СУБД, связанной с устройствами;
* jdbcStockDao: StockDao – экземпляр класса JdbcStockDao служащего для работы с частью СУБД, связанной с количеством устройств в наличии;
* CART\_SESSION\_ATTR: String – строка, хранящая в себе название сессионного аттрибута корзины.

Класс HttpSessionCartService содержит методы:

* getCart(HttpSession) : Cart – создаёт или получает значения атрибута корзины из сессии;
* addPhone(Long, Long, Cart) : void – добавляет продукт в корзину;
* addToCart(Long, Stock, Phone, Cart) : void – проверяет существует ли продукт в корзине и в зависимости от результата создает новый или обновляет существующиц;
* update(Map<Long,Long>, Cart) : void – обновляет информацию в корзине;
* checkQuantity(Long,Long) : void – проверяет количество товаров с доступным в магазине;
* remove(Long, Cart) : void – удаляет элемент из корзины;
* deleteCart(HttpSession) : void – удаляет корзину из сессии;
* findCartItem(Long, Cart) : void – найти элемент в корзине;
* calculateCart(Cart) : Order – подсчёт цены и количества продуктов в корзине.

**3.4.8 Класс** **PhoneServiceImpl**

Класс PhoneServiceImpl представляет из себя набор утилит для осуществления бизнес-логики для работы c устройствами, представленными в системе, и абстракции блока обработки запросов от блока работы с СУБД.

Класс PhoneServiceImpl содержит следующие поля:

* jdbcPhoneDao: PhoneDao – экземпляр класса JdbcPhoneDao служащего для работы с частью СУБД, связанной с устройствами;
* jdbcStockDao: StockDao – экземпляр класса JdbcStockDao служащего для работы с частью СУБД, связанной с количеством устройств в наличии;
* jdbcTemplate: JdbcTemplate – экземпляр класса JdbcTemplate, позволяющий отправлять запросы в базу данных.

Класс PhoneServiceImpl содержит методы:

* getPhoneByKey(Long) : Optional<Phone> – получает устройство по уникальному идентификатору;
* getPhoneByModel(String) : Optional<Phone> – получает устройство по названию модели;
* savePhone(Phone, Long, String) : void – сохраняет телефон и необходимую информацию в базе данных;
* getPhones(int, int) : List<Phone> – получить список устройств из базы данных с определенным смещением и ограничением;
* getPhones(ParamsForSearch) : List<Phone> – получить список устройств из базы данных с условием поиска, сортировки, определенным смещением и ограничением;
* populateFields(PhoneAddDto) : Phone – преобразование DTO-объекта в модель Phone;
* deletePhone(Long) : void – удалить устройство и связанную с ним информацию из системы;
* updateDevice(Phone, Long, String[] colors, Long) : void – обновляет информацию об устройстве в системе;
* countPhoneQuantity(ParamsForSearch) : Long – подсчёт количества устройств в системе;
* getHotPricedPhones(int, int) : List<Phone> – получить телефоны со скидкой.

**3.4.9 Класс UserServiceImpl**

Класс UserServiceImpl представляет из себя набор утилит для осуществления бизнес-логики для работы c пользователями и сотрудниками, представленными в системе, и абстракции блока обработки запросов от блока работы с СУБД.

Класс UserServiceImpl содержит следующие поля:

* jdbcUserDao: UserDao – экземпляр класса JdbcUserDao служащего для работы с частью СУБД, связанной с пользователями и сотрудниками.

Класс UserServiceImpl содержит методы:

* loadUserByUsername(String) : User– получить информацию о пользователе по его имени учётной записи;
* getUsersExceptAdmin(int, int) : List<User> – получить список пользователей кроме администратора;
* updateUserStatus(int, String) : void – обновляет статус пользователя в система;
* registerUser(String, String) : void – регистрирует пользователя в системе;
* getEmployees() : List<Employee> – получить список сотрудников магазина.

**3.5 Блок валидации запросов**

**3.5.1 Класс OrderDataDtoValidator**

Класс OrderDataDtoValidator представляет из себя набор утилит для проверки запроса пользователя для создания заказа, созданного через форму, на правильность и отсутствие каких-либо ошибок.

Класс OrderServiceImpl содержит следующие поля:

* REGEX\_PHONE: String – статическая строка, задающая регулярное выражение для проверки на валидность номера мобильного телефона.

Класс OrderServiceImpl содержит методы:

* supports(aClass: Class<?>) : boolean – выполняет проверку на возможность валидации запросов предоставленного класса;
* validate(Object, Errors) : void – валидирует запрос, при возникновении ошибок они добавляются в объект errors;
* validateWithRegex(String) : void – валидирует сроку с номером телефона с помощью регулярного выражения;

**3.5.2 Класс PhoneAddDtoValidator**

Класс PhoneAddDtoValidator представляет из себя набор утилит для проверки администраторского запроса по добавлению устройства в базу данных, созданного через форму, на правильность и отсутствие каких-либо ошибок.

Класс PhoneAddDtoValidator содержит методы:

* supports(aClass: Class<?>) : boolean – выполняет проверку на возможность валидации запросов предоставленного класса;
* validate(Object, Errors) : void – валидирует запрос, при возникновении ошибок они добавляются в объект errors.

**3.5.3 Класс PhoneArrayDtoValidator**

Класс PhoneArrayDtoValidator представляет из себя набор утилит для проверки запроса пользователя для обновления корзины, созданного через форму, на правильность и отсутствие каких-либо ошибок.

Класс PhoneArrayDtoValidator содержит следующие поля:

* jdbcStockDao: StockDao – экземпляр класса JdbcStockDao, служащий для работы с информацией о количестве устройств в базе данных.

Класс PhoneArrayDtoValidator содержит методы:

* supports(aClass: Class<?>) : boolean – выполняет проверку на возможность валидации запросов предоставленного класса;
* validate(Object, Errors) : void – валидирует запрос, при возникновении ошибок они добавляются в объект errors;
* validateWithRegex(String) : void – валидирует сроку с номером телефона с помощью регулярного выражения;

**3.5.4 Класс PhoneDtoValidator**

Класс PhoneDtoValidator представляет из себя набор утилит для проверки запроса пользователя для создания заказа, созданного через форму, на правильность и отсутствие каких-либо ошибок.

Класс PhoneDtoValidator содержит следующие поля:

* jdbcStockDao: StockDao – экземпляр класса JdbcStockDao, служащий для работы с информацией о количестве устройств в базе данных.

Класс PhoneDtoValidator содержит методы:

* supports(aClass: Class<?>) : boolean – выполняет

проверку на возможность валидации запросов предоставленного класса;

* validate(Object, Errors) : void – валидирует запрос, при возникновении ошибок они добавляются в объект errors;
* validateWithRegex(String) : void – валидирует сроку с номером телефона с помощью регулярного выражения;

**3.6 Блок обработки ошибок**

**3.6.1 Класс ResponseErrors**

Класс ResponseErrors объединяет ошибки, возникшие при создании ответа.

Класс ResponseErrors содержит следующие поля:

* errorMessage: String – строка хранящая в себе сообщение о причине ошибки.

Класс ResponseErrors содержит следующие методы:

* ResponseErrors (String)– конструктор для инициализации полей экземпляра класса;
* getErrorMessage(): String – получение значения поля errorMessage;
* setErrorMessage(String): void – установка значения в поле errorMessage.

**3.6.2 Класс ValidationErrors**

Класс ValidationErrors объединяет ошибки, возникшие при валидации запроса по добавлению в корзину.

Класс ValidationErrors содержит следующие поля:

* errors: List<ObjectError> – список возникших ошибок.

Класс ValidationErrors содержит следующие методы:

* ValidationErrors (List<ObjectError>)– конструктор для инициализации полей экземпляра класса;
* getErrors(): List<ObjectError> – получение значения поля errors;
* errors(List<ObjectError>): void – установка значения в поле errors.

**3.6.3 Класс** **GlobalController**

Класс GlobalController обрабатывает возникшие ошибки и перенаправляет пользователя на страницу с информацией.

Класс GlobalController содержит следующие методы:

* exception404 (): String – перенаправляет на стандартную страницу для отображения ошибок;
* accessDenied(): String – перенаправляет на стандартную страницу для отображения ошибок;
* internalServerError(): String – перенаправляет на стандартную страницу для отображения ошибок.

**3.7 Описание модели данных**

**3.7.1 Таблица сolors**

Таблица служит для хранения цветов устройств.

Поля:

* id – первичный ключ;
* code – название цвета.

**3.7.2 Таблица phones**

Таблица содержит информацию об устройствах, их характеристиках (модель, бренд, ссылка на изображение и т.п.).

Поля:

* id – первичный ключ;
* brand – бренд устройства;
* model – модель устройства;
* price – цена устройства;
* discountPercent – размер скидки;
* displaySizeInches – размер экрана устройства (в пикселях);
* weightGr – вес в граммах;
* lengthMm – длина в миллиметрах;
* widthMm – ширина в миллиметрах;
* heightMm – высота в миллиметрах;
* deviceType – тип устройства;
* displayResolution – разрешение экрана;
* pixelDensity – плотность пикселей;
* displayTechnology – технология дисплея;
* backCameraMegapixels – мегапиксели задней камеры;
* frontCameraMegapixels – мегапиксели передней камеры
* internalStorageGb – размер внутреннего хранилища в

гигабайтах;

* batteryCapacityMah – ёмкость батареи;
* talkTimeHours – количество часов работы при использовании;
* standByTimeHours – количество часов работы в простое;
* bluetooth – версия Bluetooth;
* imageUrl – ссылка на изображение;
* description – описание устройства.

**3.7.3 Таблица phone2Color**

Таблица служит для связи таблицы phones с таблицей colors, связь многие ко многим.

Поля:

– phoneId – внешний ключ (идентификатор устройства из таблицы phones);

* colorId – внешний ключ (идентификатор цвета из таблицы

colors).

**3.7.4 Таблица stocks**

Таблица содержит информацию о наличии устройств в магазине.

Поля:

* phoneId – внешний ключ(идентификатор устройства);
* stock – количество устройств в наличии;
* reserved – количество зарезервированных устройств.

**3.7.5 Таблица orders**

Таблица содержит информацию об заказах (цена, цена доставки) и об контактных данных покупателя(имя, фамилия, телефон, адрес)

Поля:

* id – первичный ключ;
* subtotal – цена без доставки;
* deliveryPrice – цена доставки;
* totalPrice – общая цена;
* firstName – имя покупателя;
* firstName – имя покупателя;
* lastName – фамилия покупателя;
* deliveryAddress – адрес доставки;
* contactPhoneNo ­­– номер телефона;
* additionalInfo – дополнительная информация;
* status – статус заказа;
* date – дата заказа.

**3.7.6 Таблица orderItems**

Таблица содержит информацию о элементах заказа и связана с таблицами phones и orders, связью многие к одному и один ко многим соответственно.

Поля:

* id – первичный ключ;
* phoneId – внешний ключ (идентификатор устройства);
* orderId – внешний ключ (идентификатор заказа);
* quantity – количество элементов.

**3.7.7 Таблица employees**

Таблица содержит информацию о сотрудниках.

Поля:

* id – первичный ключ;
* firstName – имя сотрудника;
* lastName – фамилия сотрудника;
* position – должность сотрудника;
* employeeType – тип сотрудника.

**3.7.8 Таблица users**

Таблица служит для хранения данных о зарегистрированных пользователях приложения, используется для аутентификации пользователей.

Поля:

* userName – имя учетной записи;
* password – закодированный пароль;
* role – роль пользователя для предоставления прав;
* isAccountNonLocked – проверки блокировки пользователя.