Декомпозиция системы

Система разделяется на следующие ключевые модули, каждый из которых отвечает за свою бизнес-логику.

1. Модуль аутентификации и авторизации (Auth Module)

Ответственность: Управление учетными записями пользователей, вход и выход в систему, разграничение прав доступа на основе ролей (`User`, `Manager`, `Administrator`).

Внутренние компоненты:

`LoginService`: Обрабатывает логин/логаут, валидирует учетные данные, выдает токен сессии.

`UserRegistrationService`: Регистрирует новых клиентов (`Client`).

`RoleService`: Управляет ролями и проверяет права доступа .

Сущности БД: `User` (id, username, password, email, role)

API Endpoints (пример):

`POST /api/auth/login`

`POST /api/auth/logout`

`POST /api/auth/register` (для клиентов)

2. Модуль управления автомобилями (Car Management Module)

Ответственность: Полное управление автопарком. Добавление, редактирование, просмотр и обновление статусов автомобилей.

Внутренние компоненты:

`CarCatalogService`: CRUD-операции для сущности `Car`. Предоставляет каталог автомобилей для клиентов.

`CarStatusService`: Отвечает за обновление статуса автомобиля (`CarStatus`), а именно ("Available", "Rented", "Under Maintenance").

`CarSearchService`: Поиск и фильтрация автомобилей по бренду, модели, цене, статусу и т.д.

Сущности БД: `Car` (vin, brand, model, year, price, mileage, status)

API Endpoints (пример):

`GET /api/cars` (каталог для клиентов)

`GET/PUT/POST /api/admin/cars` (управление для менеджеров/админов)

`PUT /api/cars/{vin}/status`

3. Модуль управления клиентами (Client Management Module)

Ответственность: Регистрация и управление профилями клиентов.

Внутренние компоненты:

`ClientProfileService`: Управление данными клиента (`fullName`, `phone`, `email`). Для регистрации создает связанную запись `User`.

Сущности БД: `Client` (clientId, fullName, phone, email, userId)

API Endpoints (пример):

`GET /api/client/profile`

`PUT /api/client/profile`

4. Модуль управления прайс-листом (Pricing Module)

Ответственность: Управление ценами на автомобили, включая базовые и рыночные цены.

Внутренние компоненты:

`PriceCalculationService`: Расчет стоимости на основе `PriceList`. Учитывает `basePrice`, `marketPrice`.

`PriceUpdateService`: Обновление цен в `PriceList` и логирование изменений.

Сущности БД: `PriceList` (priceId, carId, basePrice, marketPrice, lastUpdated)

API Endpoints (пример):

`GET /api/prices/car/{carId}` (для отображения цены в каталоге)

`POST /api/admin/prices` (для менеджеров)

5. Модуль управления договорами (Contract Management Module)

Ответственность: Оформление, валидация и управление жизненным циклом договоров покупки.

Внутренние компоненты\*\*

`ContractCreationService`: Основная логика создания договора. Связывает `Client`, `Car`, вычисляет `amount` на основе данных из `Pricing Module`.

`ContractValidationService`: Проверяет условия перед созданием договора (доступность автомобиля, валидность данных клиента).

`ContractDocumentService`: Реализует метод `generateDocument()`, формируя PDF-документ договора.

`ContractStatusService`: Управляет статусом договора (`ContractStatus`), например, "Active", "Completed", "Cancelled".

Сущности БД: `Contract` (contractId, date, clientId, carId, amount, status)

API Endpoints (пример):

`POST /api/contracts` (для менеджеров)

`GET /api/contracts/{id}`

`PUT /api/contracts/{id}/status`

6. Модуль управления платежами (Payment Module)

Ответственность: Обработка платежей по договорам.

Внутренние компоненты:

`PaymentProcessingService`: Интегрируется с платежными шлюзами (эквайринг) для приема платежей по карте и т.д. Реализует метод `process()`.

`PaymentValidationService`: Проверяет успешность транзакции, валидирует данные.

`PaymentHistoryService`: Ведет историю всех платежей.

Сущности БД: `Payment` (paymentId, contractId, amount, method, status)

API Endpoints (пример):

`POST /api/payments/process` (вызов от фронтенда при оплате)

`GET /api/payments/contract/{contractId}`

7. Модуль отчетности (Reporting Module)

Ответственность: Генерация отчетов для менеджеров и администраторов.

Внутренние компоненты:

`ReportGeneratorService`: Формирует отчеты (`Report`) различных типов (`ReportType`), например, "Финансовый отчет за месяц", "Отчет по загрузке автопарка".

`DataExportService`: Реализует метод `export()`, позволяя выгружать отчеты в форматах PDF, CSV, XLSX.

Сущности БД: `Report` (reportId, type, generatedDate, parameters, filePath)

API Endpoints (пример):

`POST /api/reports/generate` (асинхронный запуск генерации)

`GET /api/reports/{reportId}/download`

---

8. Модуль администрирования (Admin Module)

Ответственность: Функции, доступные только для роли `Administrator`.

Внутренние компоненты:

`UserManagementService`: Позволяет администратору управлять всеми пользователями системы (`manageUsers()`).

`SystemConfigurationService`: Управление настройками системы (`configureSystem()`), например, параметрами штрафов, настройками почты.

API Endpoints (пример):

`GET/PUT/POST /api/admin/users`

`GET/PUT /api/admin/system-settings`

Схема взаимодействия модулей (на примере создания договора)

1. \*Менеджер\* входит в систему через \*Auth Module\*.

2. Через \*Car Management Module\* менеджер находит нужный автомобиль и проверяет его статус.

3. Через \*Client Management Module\* менеджер выбирает клиента.

4. \* Contract Management Module\* запрашивает у \*Pricing Module\* расчет стоимости выбранного автомобиля.

5. \*Contract Management Module\* создает договор, меняя статус автомобиля на "Reserved" через \*Car Management Module\*.

6. \*Client\* производит оплату через \*Payment Module\*.

7. После успешного платежа \*Payment Module\* уведомляет \*Contract Management Module\* , который меняет статус договора на \*Active\*, а \*Car Management Module\*\* меняет статус автомобиля на "sold out".

8. В любой момент \*Manager\* может сгенерировать отчет о деятельности через \*Reporting Module\*.