

Проект: Логистична компания

Документация

 **Михаил – База данни + Архитектура**

1. Проектиране на базовата архитектура

Задачи и реализация:

- **Определяне на основните модули**

Основните модули на системата са:

- **Потребители (Users)** – базов модел за всеки, който ползва системата. Връзка към роли и опционално към клиент/служител.
 - **Роли (Roles)** – дефинират правата и достъпа на потребителите. Включват admin, courier, office, client.
 - **Служители (Employees)** – служители на компанията с позиция (office/courier) и връзка към фирмата.
 - **Клиенти (Clients)** – клиенти на компанията, с телефон и множество адреси (ClientAddress).
 - **Адреси на клиенти (ClientAddress)** – един клиент може да има няколко адреса.
 - **Пратки (Shipments)** – информация за изпратени и получени пратки, връзка към подател и получател, статус, цена, тегло.
 - **Офиси (Offices)** – физически места на компанията за получаване/предаване на пратки.
 - **Логистична компания (LogisticCompanies)** – базова информация за фирмата, връзка към служители, клиенти и офиси.
- **Избор на връзки между таблиците**
 - User **hasOne** Employee и/или Client.
 - Employee **belongsTo** User и **belongsTo** LogisticCompany.

- Client **belongsTo** User и **belongsTo** LogisticCompany.
 - Client **hasMany** ClientAddress.
 - Shipment **belongsTo** Client като sender и receiver.
 - Office **belongsTo** LogisticCompany.
 - **Подготовка на ER диаграма**
 - Визуализира всички модули и връзките между тях.
 - Обозначава foreign keys и cardinality (1:1, 1:N).
 - ER диаграмата е ключова за лесна навигация на данните и предотвратяване на referential integrity грешки.
-

2. Създаване на база данни

Реализация:

- **Laravel Migrations**
 - За всяка таблица се създава отделен migration файл.
 - Миграциите дефинират структурата, типа на колоните, индексите и връзките.
 - Пример: 2026_01_20_000700_create_users_table.php създава таблицата users с foreign key към roles.
 - **Foreign Keys и индекси**
 - Всяка таблица с връзка към друга таблица използва foreignId()->constrained()->onDelete('cascade').
 - Осигурява referential integrity – ако се изтрие родител, се изтриват зависимите записи.
 - **Soft Deletes**
 - Таблиците като clients, employees, logistic_companies, shipments използват softDeletes().
 - Позволява възстановяване на данни, предотвратява загуба на критична информация.
-

3. Таблицы и полета

Таблица	Полета	Връзки	Допълнителни бележки
users	id, name, email, profile_image, password, role_id, timestamps	belongsTo Role	Основен модел за authentication
roles	id, name, timestamps	hasMany Users	Роли: admin, courier, office, client
employees	id, user_id, company_id, position, timestamps, deleted_at	belongsTo User, LogisticCompany	Дефинира служители на фирмата
clients	id, user_id, company_id, phone, timestamps, deleted_at	belongsTo User, LogisticCompany; hasMany ClientAddress	Могат да имат множество адреси
client_addresses	id, client_id, city, address, timestamps, deleted_at	belongsTo Client	Множество адреси за един клиент
shipments	id, sender_id, receiver_id, weight, price, status, timestamps, deleted_at	belongsTo Client (sender/receiver)	Информация за пратки и техния статус
offices	id, company_id, name, address, timestamps	belongsTo LogisticCompany	Офиси за предаване и получаване
logistic_companies	id, name, address, timestamps, deleted_at	hasMany Employees, Clients, Offices	Основна фирмена структура



Мартин – Backend: Потребители, роли, сигурност

1. Регистрация и вход на потребители

Реализация:

- UserController

- Метод `register()` създава нов потребител и асоциира роля.
 - Метод `login()` проверява имейл и парола, връща JWT токен или сесия.
 - Хеширане на пароли се извършва с `Hash::make($password)` за сигурност.
 - **Валидация:**
 - Използване на `FormRequest` за проверка на входящите данни.
 - Проверка за уникалност на имейл, задължителни полета, минимална дължина на паролата.
 - Гарантира, че данните въвеждани от потребителя са валидни преди запис в базата.
-

2. Авторизация и роли

Реализация:

- **Роли:**
 - Ролите са дефинирани в таблицата `roles` – `admin`, `office`, `courier`, `client`.
 - `hasRole` проверява текущия `user` и дали има достъп до дадена функционалност.
 - **Права на достъп:**
 - **Admin** – пълен достъп до всички CRUD операции и справки.
 - **Office** – достъп до регистриране на пратки и преглед на клиентите в компанията.
 - **Courier** – достъп до пратки, които трябва да достави.
 - **Client** – достъп само до собствените си пратки и адреси.
-

3. CRUD за потребители

Реализация:

- **Служители и клиенти:**
 - **Create:** Създаване на `user`, присвояване на правилната роля, създаване на свързан `employee` или `client` запис.

- **Read:** Извличане на списък с users, филтриране по role и статус.
 - **Update:** Актуализиране на информация и промяна на роли.
 - **Delete:** Soft delete, за да може възстановяване на данни при нужда.
 - **Връзки:**
 - user->employee и user->client се проверяват чрез Eloquent relationships.
 - Unit тестове гарантират, че тези връзки работят правилно.
-

4. API за потребители

Endpoints и функционалности:

- **Регистрация:** POST /api/register
 - Създава нов потребител и асоциира роля.
 - Връща успешно създаден user и токен/сесия.
 - **Вход:** POST /api/login
 - Проверява имейл и парола.
 - Връща JWT токен за frontend.
 - **Смяна на роли:** PATCH /api/users/{id}/role
 - Достъп само за администратор.
 - Позволява промяна на ролята на потребител.
 - **CRUD за служители и клиенти:**
 - GET /api/users – списък на потребители с филтри.
 - POST /api/users – създаване.
 - PUT /api/users/{id} – актуализация.
 - DELETE /api/users/{id} – soft delete.
-

5. Валидация и сигурност

- **FormRequest validation:**

- Email: уникален, валиден формат.
 - Парола: минимум 8 символа.
 - Role: трябва да съществува в таблицата roles.
 - **Password hashing:**
 - Никога не се записват пароли в plain text.
 - **За роли:**
 - Предотвратява достъп до endpoints, които потребителят няма право да използва.
-

Ростислав – Backend: Основни функционалности + справки

1. CRUD за логистична компания, офиси и пратки

Реализация:

- **Логистична компания (LogisticCompanyController)**
 - CRUD операции: създаване, редакция, изтриване (soft delete) и извличане на фирми.
 - Soft delete позволява възстановяване на случайно изтрети фирми.
 - Връзка с employees и offices за управление на персонал и локации.
- **Офиси (OfficeController)**
 - CRUD за офисите на компанията.
 - Полета: име, адрес, телефон.
 - Връзка към logistic_companies чрез foreign key.
- **Пратки (ShipmentController)**
 - Полета: sender (user/клиент), receiver (user/клиент), тегло, цена, статус, registered_by (служител).
 - Soft deletes за безопасност при изтриване.
 - CRUD операции: създаване на пратка от служител, промяна на статус, изтриване и извличане на пратки.

2. Логика за пратки

Реализация:

- **Цена на пратка:**

```
$price = $weight * $rate_per_kg;  
if ($delivery_type == 'office') {  
    $price += $office_fee;  
} else {  
    $price += $address_delivery_fee;  
}
```

- Различни тарифи за доставка до офис и до адрес.
- Цена динамично се изчислява при създаване на пратка.
- **Регистрация на пратки:**
 - registered_by записва кой служител е регистрирал пратката.
 - Свързване с sender и receiver чрез belongsTo.
- **Статуси на пратки:**
 - pending – създадена, но не доставена.
 - delivered – успешно доставена.
 - cancelled – отменена.
 - Статусите се управляват чрез enum или array в модела Shipment.

3. Справки

Реализирани справки:

- **Всички служители:**
 - Извлича се списък на служители с позиции и компании.
- **Всички клиенти:**

- Извлича се списък на клиенти с телефони и свързана фирма.
 - **Всички пратки:**
 - Включва sender, receiver, тегло, цена, статус, registered_by.
 - **Пратки по служител:**
 - Филтриране на пратки, регистрирани от конкретен служител.
 - **Неполучени пратки:**
 - Статус pending.
 - **Пратки по клиент:**
 - Изпратени от клиента (sender).
 - Получени от клиента (receiver).
 - **Приходи за период:**
 - Сумиране на price за всички пратки между start_date и end_date.
 - Може да се филтрира по компания, офис или клиент.
-

Ванина – Frontend (UI + responsive дизайн)

1. Дизайн и UI компоненти

- **CSS framework:** Използване на **Bootstrap 5** за бързо изграждане на responsive интерфейс, grid система и готови компоненти (таблици, бутони, форми, карти).
- **Blade компоненти:** Създадени са повторяеми компоненти за:
 - **Таблици** – за списъци с пратки, клиенти и служители.
 - **Форми** – за CRUD операции и филтриране.
 - **Карти (Cards)** – за справки, приходи, статистики.
 - **Alert/Notification** – за съобщения за успех или грешка.
- **Моноширинен шрифт за код и справки:** Използван **Consolas / Courier New** за блокове код и таблични стойности (например филтри и API отговори).
- **Фонове и оцветяване:** Светлосив фон (#f5f5f5) за секции с данни, за визуално отделяне на съдържанието.

- **Икони и визуални индикатори:** Използвани **Bootstrap Icons** за действия като редакция, изтриване, филтриране, за по-интуитивен интерфейс.
-

2. Екрани и интерфейси

- **Вход и регистрация:**
 - Валидация на полета (email, password, required).
 - Интерактивни съобщения за грешка (invalid credentials, missing fields).
 - Различен редирект според ролята (admin → dashboard, client → shipment list).
 - **Списък с пратки:**
 - Таблица с всички пратки, status, подател, получател, тегло, цена.
 - Филтриране чрез dropdown и дата (служител, клиент, период).
 - Цветови индикатори за статус (green – delivered, yellow – pending, red – cancelled).
 - **CRUD интерфейси за клиенти, служители, офиси, компании:**
 - Формуляри с валидация на клиентска страна (required, email, phone number).
 - Използване на модални прозорци за редакция и изтриване, за да се избегне пренасочване на страници.
 - Таблицы с възможност за сортиране и търсене (по име, дата, статус).
 - **Справки и отчети:**
 - Таблицы с всички служители, клиенти, пратки.
 - Филтриране чрез dropdown и input полета за дата.
 - Визуализация на приходи с картови блокове (cards) с цветови кодове.
 - Pagination за големи списъци.
-

3. Роли и навигация

- **Динамични менюта според ролята:**
 - Admin – пълен достъп до всички модули и справки.

- Office – достъп до пратки, клиенти, офиси, справки по служител.
 - Courier – достъп само до свои пратки.
 - Client – достъп до собствените си пратки и адреси.
 - **Blade условия:**
 - @can, @role, @if(Auth::user()->role) за контрол на видимостта на бутони и менюта.
 - **Breadcrumbs и навигация:**
 - Подобряване на UX чрез breadcrumb навигация за лесно връщане назад.
 - Highlight на текущия активен модул.
-

4. Интеграция с backend

- **Получаване на данни:**
 - Blade templates получават данни директно от контролерите (compact(...)).
 - AJAX за динамични филтри и обновяване на таблици без презареждане (по избор).
 - **Форми и валидация:**
 - Laravel validation errors показвани директно в UI (\$errors->first('field')).
 - HTML5 валидация за полета като email, required, number.
 - **UI feedback:**
 - Flash messages за успех и грешка (session('success'), session('error')).
 - Tooltip-и и икони за пояснения.
-

5. Responsive дизайн

- Използване на **Bootstrap grid system** за адаптивност:
 - Таблици и карти се скриват или променят размера при мобилни устройства.
 - Form input полета и dropdown менюта се подреждат вертикално на малки екрани.

- Media queries и класове (.d-none .d-md-block) за специфично съдържание.
 - Тестване на интерфейса на различни устройства (desktop, tablet, mobile).
-

6. Добри практики и допълнителни оптимизации

- Моноширинен шрифт за код и числа, за по-добра четливост.
 - Използване на Blade components и partials за повторяеми UI елементи.
 - Консистентни цветове и стилове за статуси, бутони и таблици.
 - Лек фон за секции с данни и code-like блокове (#f5f5f5) за визуално отделяне.
 - Accessibility: контрастни цветове, описателни бутони и labels за screen readers.
-



Ива – Документация + Тестване + Мениджмънт

1. Документация

- **Описание на всички модули и функционалности**
 - Подробно описание на всеки модул:
 - **Потребители и роли** – CRUD операции, регистрация, login, role-based access control.
 - **Клиенти и служители** – връзка към логистични компании, addresses, profile info.
 - **Пратки** – създаване, редакция, статуси (pending, delivered, cancelled), цена и тегло.
 - **Офиси** – адреси на офисите, връзка към логистични компании.
 - **Логистични компании** – основна информация, свързани офиси, служители и пратки.
 - **Справки** – всички пратки, пратки по служител, пратки от/до клиент, неполучени пратки, приходи за период.
- **ER диаграма и обяснения на таблиците**
 - ER диаграмата показва всички връзки (hasOne, hasMany, belongsTo) между таблиците.

- Всеки модел е описан с неговите полета, типове и връзки.
 - Обяснено е защо се използват foreign keys и soft deletes.
 - **Примери на екрани и инструкции за ползване**
 - Скриншоти на формуляри за CRUD операции.
 - Примери на справки и филтрирани таблици.
 - Пояснения за различните роли и достъп до функционалности.
-

2. Тестване

- **Unit тестове за модели**
 - **UserTest** – създаване, update, delete, връзки с Employee и Client.
 - **ClientTest** – addresses, sentShipments, receivedShipments.
 - **EmployeeTest** – връзка с логистична компания, пратки.
 - **ShipmentTest** – статуси, цена, тегло, регистриран от служител.
 - **LogisticCompanyTest** – CRUD, връзки към офиси, служители и пратки.
- **Тестове за отношения**
 - Проверка на hasOne, hasMany и belongsTo.
 - Уверяване, че връзките връщат правилните записи.
 - Тестове за soft deletes и restore.
- **Тестове за справки и филтри**
 - Филтриране на пратки по:
 - Служител
 - Клиент (изпратени/получени)
 - Период (start_date, end_date)
 - Проверка на сумарни приходи и статистики.
- **Тестови данни**

- Използване на Laravel **Factories** и **Seeders** за подготовка на тестова база данни.
 - Изпълнение на тестовете върху **SQLite in-memory database**, за бързо и чисто тестване.
-

3. QA и Мениджмънт

- **Проследяване на бъгове и решаването им**
 - Регистриране на грешки в трекер (примерно Jira или Trello).
 - Свързване на грешките с конкретни модули, за да се идентифицират причините.
 - Потвърждаване на корекции чрез повторно тестване.
 - **Координация на екипа**
 - Разпределяне на задачи между Frontend и Backend разработчици.
 - Проверка на прогреса на документацията, тестовете и базата данни.
 - Осигуряване, че всички зависимости (Roles, Factories, Seeders) са налични за unit тестовете.
 - **Финално оформление на проекта**
 - Подготовка на крайна документация в Word/Markdown/PDF.
 - Добавяне на инструкции за setup и стартиране на системата.
 - Описание на структурата на базата, ER диаграмата и всички интерфейси.
 - **Валидация на формуляри и филтри**
 - Проверка дали всички полета във frontend формите са валидирани както на клиентската страна (HTML5), така и на backend (Laravel validation).
 - Тестове за input edge cases (празни стойности, невалидни данни, специални символи).
 - Потвърждение, че грешките се показват ясно на потребителя.
-



1. Управление на изключения

- **Централизирано чрез Handler**

- Всички необработени грешки се прихващат в `app/Exceptions/Handler.php`.
- Превръщане на PHP exceptions в JSON отговори за API или показване на потребителски съобщения за frontend.
- Логване на грешки чрез `Log::error()` за проследяване на проблеми.

- **Специфични try/catch блокове**

- При критични операции (например създаване на пратки, присвояване на роли) се използват try/catch блокове за по-добър контрол на поведението.
- Връщане на детайлна информация за грешка (validation failed, foreign key missing, duplicate entry).

- **Validation**

- Laravel FormRequest класове за всеки модул (UserRequest, ClientRequest, ShipmentRequest).
 - Проверка на задължителни полета, формати (email, phone, numeric, min/max).
 - Централизирана обработка на validation exceptions чрез ValidationException.
-

2. Вертикална структура

- **Модулно разделение на кода**

- **Models** – дефиниране на всички Eloquent модели с връзки (hasOne, hasMany, belongsTo).
- **Controllers** – логика за CRUD, API endpoints, справки.
- **Migrations** – създаване на таблици, foreign keys, soft deletes, индекси.
- **Factories** – създаване на фиктивни данни за unit тестове.
- **Tests** – Unit тестове за всеки модел, функционални тестове за API и връзки.
- **Seeders** – попълване на базата с начални роли, тестови потребители, компании, клиенти и пратки.

3. Валидация

- **Backend**

- FormRequest класи за всички модули с правила за валидация на входните данни.
- Примери:
- 'email' => 'required|email|unique:users,email',
- 'phone' => 'required|string|max:20',
- 'weight' => 'required|numeric|min:0.1',
- Връщане на структурирани JSON грешки за API.

- **Frontend**

- HTML5 validation (type="email", required, pattern).
- JavaScript проверки за динамични форми (например проверка на тегло и адрес при създаване на пратка).
- Филтриране на forms преди изпращане към backend за по-добро UX.

- **4. Unit тестове**

- **Моделни тестове**

- Проверка на hasOne, hasMany, belongsTo.
- Тестване на soft deletes и restore.
- Примери:
 - User – връзки с Employee и Client.
 - Client – addresses, sentShipments, receivedShipments.
 - Shipment – статуси, цена, тегло, регистриран от служител.

- **Функционални тестове**

- CRUD операции за всички критични модули.

- Проверка на business логика: price calculation, shipment status changes, access control.
 - **Laravel Factories**
 - Автоматично генериране на фиктивни потребители, клиенти, служители, компании и пратки.
 - Лесна подготовка на данни за unit тестове без ръчно попълване.
-

5. Security context

- **Authentication & Authorization**
 - Laravel auth scaffolding (или custom JWT) за проверка на потребители.
 - Role-based access control – различни права за admin, courier, office, client.
- **Password hashing**
 - Всички пароли се хешират с Hash::make.
- **CSRF защита**
 - Включена по подразбиране в Laravel форми.
- **Soft deletes**
 - Възможност за възстановяване на потребители, клиенти, пратки.
 - Предотвратява случайна загуба на данни и увеличава сигурността на данните.
- **Validation & Exception handling**
 - Проверка на всички input данни.