Нижче наведено кілька ідей та рекомендацій щодо організації роботи з даними клієнтів і їхніх автомобілів, а також щодо взаємодії з державними базами та страховими компаніями. Окрему увагу приділено питанню зберігання великих обсягів даних.

**1. Синхронізація з державними базами**

1. **Чи потрібна ця інтеграція?**
   * Якщо для вашого СТО важливе офіційне підтвердження технічних даних авто (наприклад, перевірка VIN, поточної реєстрації, історії ДТП), то синхронізація з державною базою може бути корисною.
   * Якщо ж не планується офіційного зв’язку з державними службами, можна обійтись внутрішнім зберіганням інформації, яку клієнт надає самостійно.
2. **Можливі складнощі**
   * Отримання офіційних доступів або ліцензій на інтеграцію з держреєстрами.
   * Безпека даних і дотримання юридичних вимог щодо захисту персональної інформації (GDPR або локальне законодавство).
   * Різниця у форматах (кожна країна може мати свої специфічні API чи відсутність API взагалі).
3. **Підхід “на майбутнє”**
   * Якщо не впевнені, чи потрібна ця опція, можна **спроєктувати систему з можливістю додавання інтеграції пізніше**.
   * Наприклад, тримати поле “vinStatus” чи “govSyncStatus” у моделі авто, але використовувати його лише тоді, коли інтеграція з’явиться.

**2. Робота зі страховими компаніями**

1. **Сценарії взаємодії**
   * Клієнт потрапляє в ДТП, має поліс страхування й приходить до СТО з метою отримати покриття ремонту.
   * СТО ініціює запит до страхової компанії або заповнює онлайн-форму з деталями пошкоджень та кошторисом.
   * Страхова компанія має підтвердити суму виплат.
2. **Переваги інтеграції**
   * Автоматизоване виставлення рахунку: дані про замовлення й кошторис (запчастини, робота) можуть одразу передаватися у страхову.
   * Прискорення процедури відшкодування вартості ремонту (не потрібно дублювати інформацію в кількох системах).
3. **Організація інтеграції**
   * **REST API страхової**: якщо страхові компанії надають API, можна налагодити автоматизоване надсилання даних про ремонт.
   * **Експорт PDF**: якщо API немає, можна сформувати PDF-звіт (кошторис, опис пошкоджень) і надсилати його у страхову електронною поштою.
   * **Розділ “Страхова інформація”** у системі: зберігати назву страхової, номер поліса, термін дії, контактні дані для звернення.

**3. Зберігання великих обсягів даних**

1. **Мінімізація зберігання**
   * Зберігати лише необхідний мінімум: VIN, марка/модель, рік випуску, історія ремонту.
   * Якщо клієнт додає великі файли (відео з несправностями, скани документів), краще відвантажувати їх у **хмарне сховище** (наприклад, AWS S3, Google Cloud Storage чи інший провайдер). У MongoDB зберігати лише посилання на ці файли.
2. **Платна опція для додаткового обсягу**
   * Безкоштовно даєте певний ліміт (наприклад, 500 МБ) на зберігання мультимедійних даних (фото, відео).
   * Якщо клієнт (або СТО) хочуть зберігати більше матеріалів, пропонуєте додатковий платний пакет.
   * Модель монетизації: періодична абонплата або разова оплата за кожний додатковий гігабайт.
3. **Оптимізація файлів**
   * Перед завантаженням можна автоматично **зменшувати роздільну здатність зображень**, щоб не перевантажувати сховище.
   * За потреби: конвертувати відео в більш “легкий” формат (наприклад, MP4 з оптимальною компресією). Це можна робити асинхронно, використовуючи хмарні служби для обробки відео.

**4. Приклад зберігання даних у моделях (MongoDB)**

js

CopyEdit

// models/Car.js

const mongoose = require('mongoose');

const carSchema = new mongoose.Schema({

ownerId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },

make: String, // Марка

model: String, // Модель

vin: String, // VIN-код

year: Number, // Рік випуску

insuranceInfo: {

companyName: String, // Страхова компанія

policyNumber: String, // Номер поліса

validUntil: Date // Термін дії

},

// тощо

});

module.exports = mongoose.model('Car', carSchema);

js

CopyEdit

// models/FileAttachment.js

const mongoose = require('mongoose');

const fileAttachmentSchema = new mongoose.Schema({

carId: { type: mongoose.Schema.Types.ObjectId, ref: 'Car' }, // Прив'язано до певного авто або замовлення

url: String, // Посилання на файл у хмарі (наприклад, AWS S3)

fileType: String, // Тип (image, video, pdf тощо)

originalName: String // Початкова назва файлу

});

module.exports = mongoose.model('FileAttachment', fileAttachmentSchema);

**Пояснення**

* Створюємо окрему модель FileAttachment, де тримаємо інформацію про кожен файл.
* У полі url зберігаємо посилання на файл у хмарному сховищі.

**5. Підсумки та рекомендації**

1. **Інтеграція з держреєстрами**:
   * Варто зробити модуль “на майбутнє” — тобто у базі даних і коді закласти можливість зберігати поля для державних статусів, але не обов’язково реалізовувати це відразу.
2. **Співпраця зі страховими**:
   * Може суттєво спростити процес відшкодування для клієнтів.
   * Почати можна з простого сценарію — формування PDF-актів для відправки в страхову. Якщо з’являться партнери зі своїм API, підключити його.
3. **Великі обсяги даних**:
   * Зберігати в хмарному сховищі, не в колекціях MongoDB.
   * Можна реалізувати платний пакет для збільшеного ліміту зберігання файлів.
4. **Архітектурна гнучкість**:
   * Проектуйте систему так, щоби можна було легко підключити додаткові функціональні модулі (запити до держреєстрів, перевірка поліса, автоматичні звернення до страхової).
   * Поки що ці блоки можуть бути “заглушками” або вмикатися за потреби.