



Софийски университет „Св. Кл. Охридски“

Факултет по математика и информатика

Катедра „Изчислителни системи“

ДИПЛОМНА РАБОТА

на тема

**„Създаване на приложение за управление
на флотилия от превозни средства“**

Дипломант: Моника Мариова Спасова

Специалност: Електронен бизнес и електронно управление

Факултетен номер: 25600

Научен ръководител:

Гл.ас. Д-р Инж. Галя Новакова

София, 2019г.

Абстракт:

В Европа, най-големият автомобилен пазар, почти всеки два от три нови автомобили се продават на корпорации. Впоследствие повечето от тези превозни средства се регистрират като фирмени автомобили, чието обслужване компаниите целят да оптимизират. Пред висок процент от съществуващите системи за управление на автопаркове стоят ограничения при интеграцията с превозните средства. Към тяхното премахване е насочено приложението, предмет на магистърската теза. Целта на настоящата дипломна работа е да се създаде удобна и лесна за използване уеб система за управление на флотилия от превозни средства, съвместима с АЕМ/АЕМР стандарта за единен формат на телематичните данни.

Декларация за авторство:

Аз, Моника Мариова Спасова, заявявам, че тази дипломна работа е резултат от моя собствен труд под ръководството на Гл.ас. Д-р Инж. Галя Новакова. Всички използвани ресурси са упоменати в секцията „Използвана литература”. Всички чужди източници са цитирани, като се предоставя референция към източниците в секцията „Използвана литература”.

Съдържание

1. Увод.....	6
1.1. Актуалност на проблема и мотивация.....	6
1.2. Цел и задачи на дипломната работа.....	8
1.3. Очаквани ползи от реализацията.....	9
2. Преглед на областта управление на флотилия от превозни средства.....	10
2.1. Основни дефиниции.....	10
2.1.1. Управление на флотилия от превозни средства	10
2.1.2. Софтуер за управление на флотилия от превозни средства.....	10
2.1.3. Компания за управление на флотилия от превозни средства.....	10
2.1.4. Свързано превозно средство.....	10
2.1.5. Телематика.....	10
2.1.6. Автомобилна телематика.....	10
2.1.7. Обща стойност на собствеността.....	11
2.1.8. GPS.....	11
2.1.9. GPRS.....	11
2.1.10. ECU.....	11
2.1.11. CAN.....	11
2.1.12. BSP.....	11
2.2. Принцип на работа на телематиката.....	11
2.2.1. Система за глобално позициониране.....	11
2.2.2. Принцип на работа на телематиката.....	12
2.2.3. Основни компоненти на автомобилната телематична система.....	14
2.2.4. AEM/AEMP стандарт.....	16
2.2.5. Произход на телематично оборудване.....	17
2.2.6. Приложения на телематиката.....	18
2.3. Съществуващи решения.....	19
2.3.1. Autosist.....	19
2.3.2. Maintenance PRO WEB.....	21
2.3.3. FLEETIO.....	23
2.3.4. Изводи.....	26
3. Използвани технологии.....	26
3.1. Изисквания към технологиите.....	26
3.2. Използвани технологии.....	26
3.2.1. Microsoft Enity Framework.....	27
3.2.2. ASP.NET Web API 2.....	27
3.2.3. ASP.NET Identity.....	27
3.2.4. Quartz.....	28
3.2.5. Alt.js.....	29
3.2.6. ReactJS.....	29
3.2.7. React-Router.....	30
3.2.8. Axios.js.....	30
3.2.9. Material-UI.....	30

3.2.10. Recharts.....	30
4. Анализ.....	31
4.1. Концептуален модел.....	31
4.2. Функционални изисквания.....	38
4.3. Нефункционални изисквания.....	40
4.4. Качествени атрибути.....	41
4.4.1. Използваемост.....	41
4.4.2. Сигурност.....	41
4.4.3. Модифицируемост.....	41
4.4.4. Мащабируемост.....	41
4.4.5. Поддръжка.....	42
4.5. Работни процеси.....	42
4.5.1. Регистрация.....	42
4.5.2. Вписване.....	42
4.5.3. Създаване на компания.....	43
4.5.4. Създаване на шофьор към компания.....	43
4.5.5. Редактиране на превозно средство към компания.....	44
4.5.6. Създаване на задача към превозно средство.....	45
4.5.7. Филтрация на пресрочени задачи към превозно средство.....	45
4.5.8. Отбелязване на задача като завършена.....	46
4.5.9. Преглед на репорт за горивото на превозно средство.....	47
5. Проектиране.....	47
5.1. Обща архитектура.....	47
5.2. Файлова структура.....	49
5.3. База от данни.....	51
5.4. Информационна архитектура.....	54
5.5. Диаграми.....	54
6. Реализация и тестване.....	58
6.1. Реализация на модулите.....	58
6.1.1. Вписване и регистрация.....	58
6.1.2. Генериране на телематични данни.....	59
6.1.3. Използване на телематично оборудване.....	60
6.1.4. Изпращане на имейли за предстоящи и просрочени задачи.....	61
6.1.5. Изчисляване на времето/километража за следваща задача и напомняне за задача.....	63
6.1.6. Модули за компании, шофьори, превозни средства и задачи.....	64
6.2. <i>Unit</i> тестване.....	64
6.2.1. <i>Unit</i> тестове на клиентската част.....	64
6.2.2. <i>Unit</i> тестове на сървърната част.....	65
6.3. Планиране на тестването.....	65
6.3.1. Тестови сценарии.....	65
6.3.2. Стъпки за тестване (<i>test cases</i>)	66
6.3.3. Анализ на резултатите от тестването.....	66
6.4. Конфигурации за локално стартиране на системата.....	67

6.5. Хостване.....	68
7. Заключение.....	68
7.1. Обобщение на изпълнението на началните цели и претенциите за оригинални резултати.....	68
7.2. Насоки за бъдещо развитие и усъвършенстване.....	69
Използвана литература.....	71
Приложение 1 Тестови сценарии.....	74
Приложение 2 Test cases.....	76

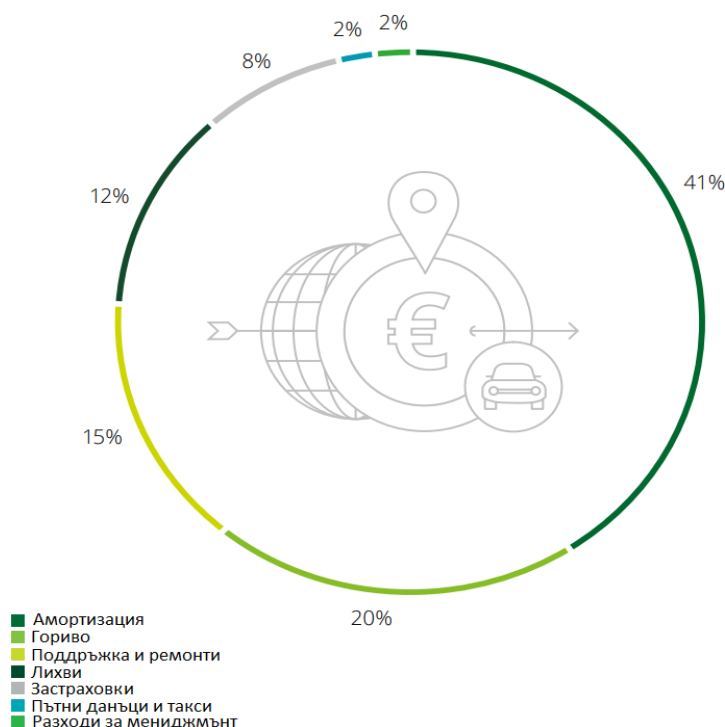
1. Увод

1.1. Актуалност на проблема и мотивация [1]

През последните години управлението на флотилия от превозни средства се превърна в индустрия за милиарди, която продължава да расте и да трупва стратегическо значение в света на променящата се мобилност. Въздействие върху този процес имат различни тенденции, които най-вероятно ще окажат съществено влияние върху бъдещето на сектора. Такива например са споделеното пътуване (споделяне на автомобила вместо неговото притежание), навлизането на автономните превозни средства в някои сектори, а също и възможността някой ден те да се движат по обществените пътища.

Спецификата на автомобилния пазар също допринася за бързия темп на развитие на системите за управление на автопаркове. Европа е най-големият автомобилен пазар в световен мащаб и в много отношения е един на най-напредналите. В него почти всички продажби на нови превозни средства са на частни или корпоративни клиенти, оставяйки много малък дял за други (например държавния сектор). В Европа почти всеки два от три нови автомобила се продават на корпорации. Впоследствие повечето от тези превозни средства се регистрират като фирмени автомобили, чието обслужване компаниите целят да оптимизират.

Ключова за идентифицирането на потенциала за намаляване на разходите е общата стойност на собствеността. На Фигура 1 е показана средната обща стойност на собствеността за превозно средство в Европа.



Фигура 1: „Средна обща стойност на собствеността на превозно средство в Европа” [1]

Вижда се, че цели 20% от стойността на собственост отиват за гориво, а 15% са предназначени за поддръжка и ремонти. Едва около 40% от общата стойност са разходи за самото превозно средство, докато останалите 60% възникват при използването му.

В подобна обстановка не е изненадващо, че компаниите търсят активно различни възможности за управление на своите автопаркове. Едни от тях се насочват към закупуване на специализиран софтуер за управление на флотилията от фирмени автомобили в границите на компанията, а други поверяват тази дейност на специализирани компании за управление на автопарка. И в двата случая в основата на успешното управление стои специализиран софтуер, разработен спрямо потребностите на бизнеса.

На пазара съществуват разнообразни софтуерни решения, предлагащи някои от следните основни категории от функции [2]:

- **Управление на превозни средства** – Управлява и рационализира процесите, задачите и събитията, свързани с превозните средства. Такива например са поддръжка, регистрация, данъчно облагане, застраховане, анализ на разходи и инвентаризация.
- **Управление на водачите** – Включва управление на информацията за водачите, записване на наказателни точки и нарушения, съставяне на графици.
- **Управление на инциденти** – Занимава се с управление на глоби и злополуки, както и разпределяне на разходите.
- **Проследяване** – Включва използването на телематика за планиране на маршрута, проследяване и нотификации.

Част от съществуващите решения са силно специализирани в една или две от тези категории без изобщо да покриват останалите. Тези с пълна функционалност (обхващащи всички категории) са подходящи предимно за големи корпорации. Освен че се предлагат на внушителна цена, в малкия или средния бизнес често няма условия за използването на част от функционалностите. Така дори и бизнесът да успее да си закупи подобна система, съществува голяма вероятност да плаща за неща, които не използва.

Някои от системите (най-често извън категорията проследяване) въобще не предлагат връзка с автомобилна телематика. Повечето от тези, които го правят, работят само с оборудване на определени производители. Подобна зависимост между софтуер и хардуер силно ограничава възможността за внедряване на друг софтуер при вече монтирани телематични устройства в превозните средства или обратното.

Тези факти мотивират идеята за разработване на система за управление на флотилия от превозни средства, която е насочена към малкия и средния бизнес. Тя обхваща основни функционалности съгласно мащабите на тази категория компании като управление на превозните средства и водачите, и дейности свързани с поддръжката.

1.2. Цел и задачи на дипломната работа

Целта на настоящата дипломна работа е да се създаде удобна и лесна за използване уеб система за управление на флотилия от превозни средства, съвместима с телематично оборудване на различни производители.

Определят се следните задачи:

1. Да се проектира и разработи система за управление на флотилия от превозни средства;
2. Приложението ще бъде проектирано и разработено съгласно с потребителските очаквания и общоприетите практики за такъв вид софтуер;
3. Системата ще предоставя достъпен и интуитивен уеб интерфейс;
4. Приложението ще разполага с добре структурирана архитектура, позволяваща лесно модифициране и разширяване в бъдеще;
5. Системата ще позволява интеграция с автомобилна телематика на различни производители;
6. Приложението ще осигурява добро ниво на сигурност спрямо кибер атаки и защита на данни;
7. Системата ще има висока производителност и бързодействие;
8. За създаването на системата ще бъдат използвани актуални уеб технологии и тенденции за разработка;
9. За съхраняването на информацията ще бъдат използвани релационни бази от данни;
10. Разработка на инфраструктура за генериране на подходящи данни, заместваща автомобилната телематика;
11. Приложението ще предоставя възможност за:
 - Регистрация и вписване на потребители;
 - Управление на компания от регистриран потребител – ще включва възможности за преглед, създаване, редактиране и изтриване на компания, а също и осигуряване на достъп на други оторизирани потребители до данните за компанията, нейните превозни средства и шофьори;
 - Управление на превозни средства и шофьори – функционалности за преглед, създаване, редактиране и изтриване на автомобили и шофьори към компания от оторизиран потребител;
 - Модул за поддръжка на автомобилите – ще дава възможност за преглед, създаване, редактиране, изтриване и филтрация на дейности по поддръжката на автомобилите (например смяна на масло, ангренажен ремък и други). Въз основа на посочени от тях интервали от време или километри, те ще могат да следят приближаващите дейности по поддръжката и да получават нотификации при просрочването им;
 - Модул за преглед на основни характеристики на превозните средства в реално време и за изминал период – ще включва репорти, базирани на данните от автомобилната телематика. Например такива ще бъдат репорти за нивото на горивото и километража в реално време и за седмица назад;

1.3. Очаквани ползи от реализацията [3]

С изпълнението на поставените задачи разработеният софтуер ще допринесе за:

- По-ефективно управление на информацията за шофьорите, превозните средства и тяхното обслужване;
- Систематизирано и централизирано управление на данните за всички компоненти на флотилията от превозни средства;
- Подобряване на производителността – Намалява се възможността за човешка грешка.
- По-голям контрол върху водачите и превозните средства;
- Проследяване на различни характеристики на превозните средства в реално време и за изминал период;
- Предотвратяване на опити за злоупотреби – Следенето на нивото на горивото и километража в реално време позволяват разкриването на опити за кражба на гориво или използването на превозните средства за неслужебни цели.
- Навременно обслужване на превозните средства – Автоматизираното следене на приближаващи действия, свързани с поддръжката на автомобилите, и изпращаните нотификации намаляват възможността за пресрочване на планирани събития.
- Намаляване на разходите на гориво – Горивото е един от неизбежните разходи, който компаниите искат да понижат. Възможността за следене на различни характеристики на превозното средство (например времето на работа на двигателя при престой) позволява да се идентифицират модели на шофиране, които повишават разходите за гориво.
- Намаляване на възможността от глоби – Например при пропускане на закупуването на винетка, плащането на застраховка „Гражданска отговорност” или закъснение на технически преглед;
- Намаляване на разходите за поддръжка – Превантивната профилактика и сервизно обслужване са едни от най-важните неща в управлението на автопарк. Навременната профилактика позволява откриването на потенциални проблеми на ранен етап и намалява възможността от задълбочаване на проблемите и необходимостта от скъпо струващи ремонти впоследствие.
- Повишаване на удовлетвореността на служителите в компанията – Голяма част от данните, които преди внедряването на системата, са обработвани ръчно, ще бъдат автоматизирани.
- Оптимизация на работния процес – След внедряването на системата служителите ще могат да отделят повече време и внимание на задачи, които не могат да бъдат автоматизирани (например работата с клиенти);
- Повишаване на удовлетвореността на клиентите – Чрез намаляването на повредите на превозните средства по време на движение и възможността за отделяне на повече внимание на техните въпроси и проблеми от страна на служителите;

2. Преглед на областта управление на флотилия от превозни средства:

2.1. Основни дефиниции

2.1.1. Управление на флотилия от превозни средства [3]

Управлението на флотилия от превозни средства включва инструментите, технологиите и практиките, с помощта на които бизнесът централизирано управлява своите превозни средства по оптимален начин.

2.1.2. Софтуер за управление на флотилия от превозни средства [4]

Софтуерът за управление на флотилия от превозни средства е решение, което помага на компаниите и организациите да управляват, организират и координират своите превозни средства чрез централна платформа. Той управлява информация с цел подобряване на производителността, намаляване на разходите и осигуряване на съответствие с правителствените разпоредби.

2.1.3. Компания за управление на флотилия от превозни средства [1],[5]

Компанията за управление на флотилия от превозни средства притежава автомобилен парк, който след сключване на договор предоставя на друга фирма. Насочени са предимно към големи корпорации, нуждаещи се от голям автопарк. [5] Тези компании обикновено предлагат услуги през целия жизнен цикъл на превозно средство, включително покупка, финансиране, поддръжка и управление, както и препродажба на превозно средство при прекратяване на договора [1].

2.1.4. Свързано превозно средство [17]

Свързано превозно средство е превозно средство, което има устройства, свързващи го с услуги извън автомобила, включително други автомобили, дом, офис или инфраструктура.

2.1.5. Телематика [6]

Телематиката е термин, който съчетава думите „телекомуникации“ и „информатика“. Той описва широкото използване на комуникационни и информационни технологии за предаване, съхраняване и получаване на информация от телекомуникационни устройства до отдалечени обекти през мрежата.

2.1.6. Автомобилна телематика [7]

Обикновено терминът телематика се използва в контекста на автомобилната индустрия, където разнообразна информация за превозното средство се изпраща до отдалечени обекти чрез мрежата.

Тъй като терминът телематика се използва предимно в рамките на автомобилния сектор, в изложението на дипломната работа понятията телематика и автомобилна телематика ще бъдат отъждествени.

2.1.7. Обща стойност на собствеността [8]

Общата стойност на собствеността включва всички разходи, които настъпват през жизнения цикъл на определени видове активи. Тя обхваща разходите за покупка и всички разходи, свързани с тяхното притежание и използване.

2.1.8. GPS [9]

GPS (*Global Positioning System*) е радио-навигационна система, която изчислява местоположението на GPS приемници по всяко време на денонощието, при всякакви метеорологични условия, навсякъде по света.

2.1.9. GPRS [13]

GPRS (*General Package Radio Service*) е негласова, високоскоростна технология за комутиране на пакети от данни в GSM мрежи (2G и 3G комуникации).

2.1.10. ECU [14]

ECU (*Electronic Control Unit*) е устройство, което контролира една или повече електрически системи в превозното средство.

2.1.11. CAN [15]

CAN (*Controller Area Network*) е стандарт, използван предимно в автомобилите, който позволява на ECU устройствата да комуникират помежду си.

2.1.12. BSP [16]

BSP (*Board Support Package*) е основният код на дадено хардуерно устройство, чрез който то ще работи с операционната система на компютъра. BSP съдържа програма, наречена *BootLoader*, която поставя операционната система и драйверите на устройства в паметта. Съдържанието на BSP зависи от конкретния хардуер и операционна система.

2.2. Принцип на работа на телематиката [9]

2.2.1. Система за глобално позициониране [9]

Съществуването на телематика без глобалната система за позициониране е невъзможно. Поради това, за да бъде разбрана работата на телематиката, е необходимо да се познава начинът, по който функционира GPS.

GPS технологията е разработена от американските военни сили за разпознаване на позицията на войниците, техните превозни средства и врагове и за наблюдение на движенията им. Към момента това е единствената напълно функционираща сателитна навигационна система.

Системата включва:

- Сателитна мрежа – Система от над 20 спътници със соларно захранване;
- Наземни комуникационни станции – Използват се за следене дали спътниците не изменят своята орбита на движение;
- Приемници;

Чрез нея могат да се определят географската ширина и дължина на приемника на Земята чрез изчисляване на времевата разлика за достигане на сигнали от различни спътници до приемника. Приемникът изчислява позицията си чрез измерване на разстоянието от него до поне три спътника. Измерването на закъснението между предаването и приемането на всеки радио сигнал позволява изчисляването на разстоянието до всеки спътник, защото сигналът се движи с известна скорост. Чрез използването на четвърти спътник може да бъде пресметната и надморската височина на приемника.

Ако превозното средство не е фабрично оборудвано с GPS, съществуват следните възможности за инсталация впоследствие:

- **Преносим GPS** – GPS приемникът е монтиран върху таблото на превозното средство и се захранва от запалката на автомобила. Тази опция е евтина и позволява лесно демантиране.
- **Оригинално заводско оборудване** – Производителите на автомобили предлагат допълнителна инсталация на GPS за онези превозни средства, които нямат фабрична такава. Основната полза от това решение е, че инсталацията е съгласно фабричен стандарт.
- **GPS устройства от производителите** – Редица производители доставят GPS устройства, които може да бъдат трайно интегрирани в превозното средство. Подходящо място за такава инсталация е слотът за радио или CD устройство. Неговите предимства са по-добрият визуален вид в сравнение с преносимото устройство и по-ниската цена от тази на фабрично инсталирания GPS.

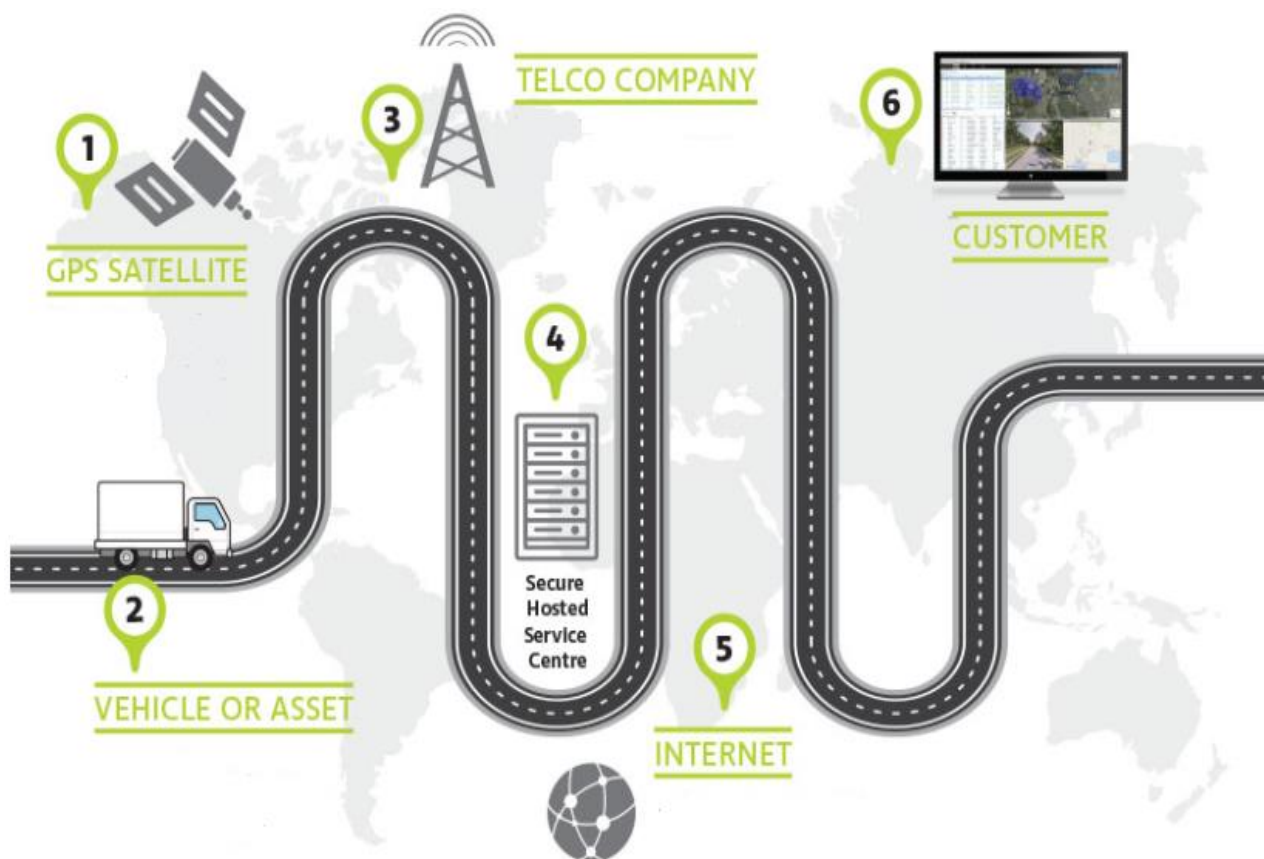
2.2.2. Принцип на работа на телематиката [9]

Трудно е да се намери универсално определение за термина телематика. Дефинициите варират в зависимост от източника на информация. Може би една от най-простите дефиниции е тази на Денис Фой, който разглежда понятието изцяло в контекста на автомобилната индустрия. Според него телематиката не е нищо повече от предаването на полезна информация от и към превозното средство.

С помощта на тази кратка дефиниция лесно се забелязва и разликата между телематични и навигационни системи. Телематичните системи използват GPS технологията за предоставяне на разнообразни услуги. Навигацията е просто една от тези услуги.

Телематичните системи работят на следния принцип (Фигура 2) [6], [9], [10]:

- **Сателит** - Позицията на превозното средство се изчислява чрез GPS приемника на телематичното устройство, монтирано в автомобила. Това се извършва от устройството за управление (*Telematics Control Unit - TCU*), което представлява централната платформа на телематичната системата в превозното средство. В него са интегрирани всички технологии по отношение на телематиката.
- **Превозно средство** – GPS координатите заедно с останалата информация за превозното средство се предават под формата на пакети към кулите на телекомуникационната компания чрез безжична връзка (например 4G, GPRS или алтернативно чрез сателитни комуникации).
- **Телекомуникационната компания** – Телекомуникационната компания събира тези данни и ги предава на главния център за услуги, който се състои от сървъри с високо ниво на сигурност с уеб хостинг. Те съхраняват и обработват данните, превръщайки ги в използвана информация.
- **Интернет** – Чрез интернет информацията от главния център за услуги достига до различните доставчици. Посредством техния програмен интерфейс данните могат да бъдат достъпни от различни системи.
- **Потребители** – Крайният потребител достига до информацията чрез уеб, мобилно или десктоп приложение.



Фигура 2: „Принцип на работа на телематиката” [10]

2.2.3. Основни компоненти на автомобилната телематична система [11]

Устройството за управление на телематиката (*Telematics Control Unit – TCU*) е централният компонент на телематичната система, управляваща множество функции като:

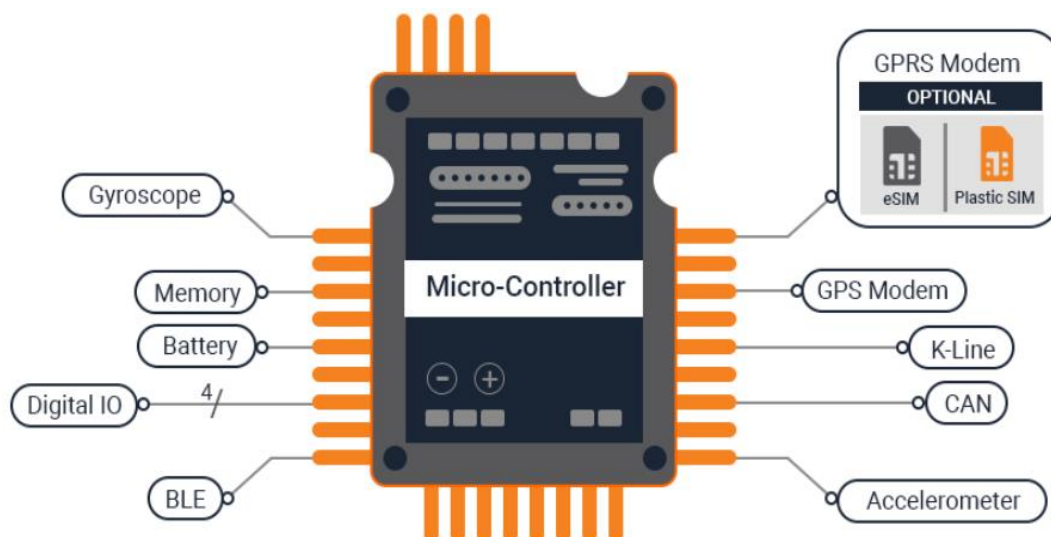
- Събиране на данни за превозното средство през CAN-BUS порта;
- Управление на информацията, събрана чрез различни комуникационни интерфейси;
- Управление на паметта и захранването;
- Управление на двустранната комуникация с главния център за услуги;
- Управление на комуникацията с потребителския интерфейс на устройството;

Основните компоненти на хардуерната архитектура на устройството за управление на телематиката са (Фигура 3):

- **GPS модул;**
- **Централен процесор** – Той има възможност за управление на паметта и за обработване на данни. Наличните в търговската мрежа телематични системи са базирани на микроконтролери и микропроцесори. За разработването на телематични системи с основни функционалности се използват предимно *Android*

базирани процесори. За създаването на модерни телематични продукти с дисплей се интегрират процесори, основани на *Linux*.

- **Модул CAN Bus** – Той управлява цялата комуникация с ECU на превозното средство. Много от наличните устройства поддържат и други интерфейси. Устройството за управление на телематиката комуникира с ECU на автомобила чрез шината CAN и извлича важна информация като работа на двигателя, скорост на превозното средство, данни от сензорите за измерване на налягането на гумите и други. Телематичната система може също да използва *K/Line* шина, за да предупреди собственика при опит за кражба (чрез нотификация, че автомобилът е запален) или да осигури възможност за дистанционно заключване и отключване.
- **Памет** – Необходима е за съхраняване на информацията по време на ненадеждна или липсваща връзка, или когато трябва да се съхраняват данни за превозното средство за бъдеща употреба. Ползена е и за поддържане на модерни телематични функции като разпознаване на реч. Флаш памет и динамичната памет с произволен достъп са често използвани в телематичните продукти.
- **Комуникационни интерфейси** – Използват се за поддържане на широка гама от комуникации, включително безжична, клетъчна и други;
- **Модул GPRS** – В някои случаи дори се използва гласова комуникация с отдалечени устройства. Често GPRS модулът идва със SIM карта в допълнение към GPRS модема. Това позволява комуникация с отдалечени устройства през клетъчната мрежа.
- **Вградена батерия** – Тя има напрежение от 3.2 до 3.4 волта за интегрирано управление на захранването. Системата на батерията е полезна за проследяване и възстановяване на откраднатото превозно средство, когато автомобилът е изключен.
- **Bluetooth модул** – за свързване на близките устройства, за взаимодействие с мобилния телефон на потребителя за разговори със свободни ръце, текстови съобщения и други;
- **Микрофон с аудио интерфейс за активиране на функции като разговори със свободни ръце и гласови команди** – Той също така поддържа стерео изход за възпроизвеждане на медийни файлове от аудио системата на автомобила.
- **Система за общ входен/изходен интерфейс за свързване на светлини и бутони** – Тя включва както аналогови, така и цифрови интерфейси.
- **Потребителски интерфейс** – За показване на важна информация като навигационни карти, скорост на превозното средство, използване на гориво и други. Потребителят може да използва интерфейса за достъп до функции като разговори със свободни ръце, преглед на карти, възпроизвеждане на медийни файлове.



Фигура 3: „Архитектура на TCU” [11]

Автомобилното телематично оборудване се предлага със следния набор от софтуерни компоненти:

- Софтуерът за стартиране *Bootloader*;
- Операционна система в реално време и BSP модули;
- Фреймуърк, който помага на приложенията да получат достъп до основните телематични функции като GPS данни.
- Софтуер за глобална навигационна спътникова система за проследяване на превозни средства в реално време и идентификация на географското местоположение.
- Софтуер за интегриране на анализи на данни за поддръжката на превозното средство, използването на гориво, моделите на използване на превозното средство и други;
- Софтуер за драйвери за мултимедийни устройства;
- Софтуер за управление на устройството за дистанционно обновяване;
- Алгоритми за многостепенна сигурност на приложението при удостоверяване на потребители, криптиране на данни, филтриране на съдържанието и други;

2.2.4. AEM/AEMP стандарт [18]

Стандартът за телематика на Асоциацията на производителите на оборудване (*Association of Equipment Manufacturers - AEM*) и Асоциацията на професионалистите по управление на оборудването (*Association of Equipment Management Professionals - AEMP*) е глобален стандарт, одобрен от Международната организация по стандартизация (*International Organization for Standardization - ISO*) през 2016г.

Той позволява на ползвателите на телематично оборудване да получават необходимите им данни в общоприет вид. Това е възможно благодарение на въвеждането на

стандартизиран XML формат, чрез който приложенията на различните доставчици на телематика да комуникират със системите за управление на автопаркове. В стандарта е заложена версия 1.0 на XML с кодиране UTF-8. Той съдържа подробна информация за наименованието и типа на всяко от полетата в XML документа.

Ако доставчиците интегрират АЕМ/АЕМР стандарта в своите приложения, значително ще улеснят работата на системите, използващи данни от оборудване на различни производители. Без наличието на подобен стандарт при използването на телематика от различни производители, информацията от всеки производител се изпраща в различен формат. Така се създава зависимост между производителите на оборудване и софтуерните решения, които използват техните данни. Усложняват се и функциите на системите, интегрирани с повече доставчици на телематични услуги, тъй като в тях се прилага различна обработка на данните спрямо техния произход.

2.2.5. Произход на телематично оборудване [12]

Съществуват два вида телематични системи според техния произход:

- **От производителите на оригинално оборудване за автомобили:**

Предимства:

- Не изискват ръчна инсталация;
- Включени са в гаранцията на автомобила;

Недостатъци:

- Работят само с този производител на оригинално оборудване, което често води до използването на различни дефиниции по отношение на данните;
- По-бавно обновяване на софтуера;
- Обикновено са специфични за модела автомобил;
- Не е възможна смяна на източника на телематични услуги;

- **От странични производители на телематично оборудване:**

Предимства:

- Има собствена гаранция;
- Работят с различни марки и модели превозни средства;
- По-бързо обновяване на софтуера;
- Възможна е смяна на източника на телематични услуги;

Недостатъци:

- Инсталират се ръчно;
- Някои устройства не могат да комуникират с ECU на превозното средство и осигуряват по-ограничено разнообразие от данни;

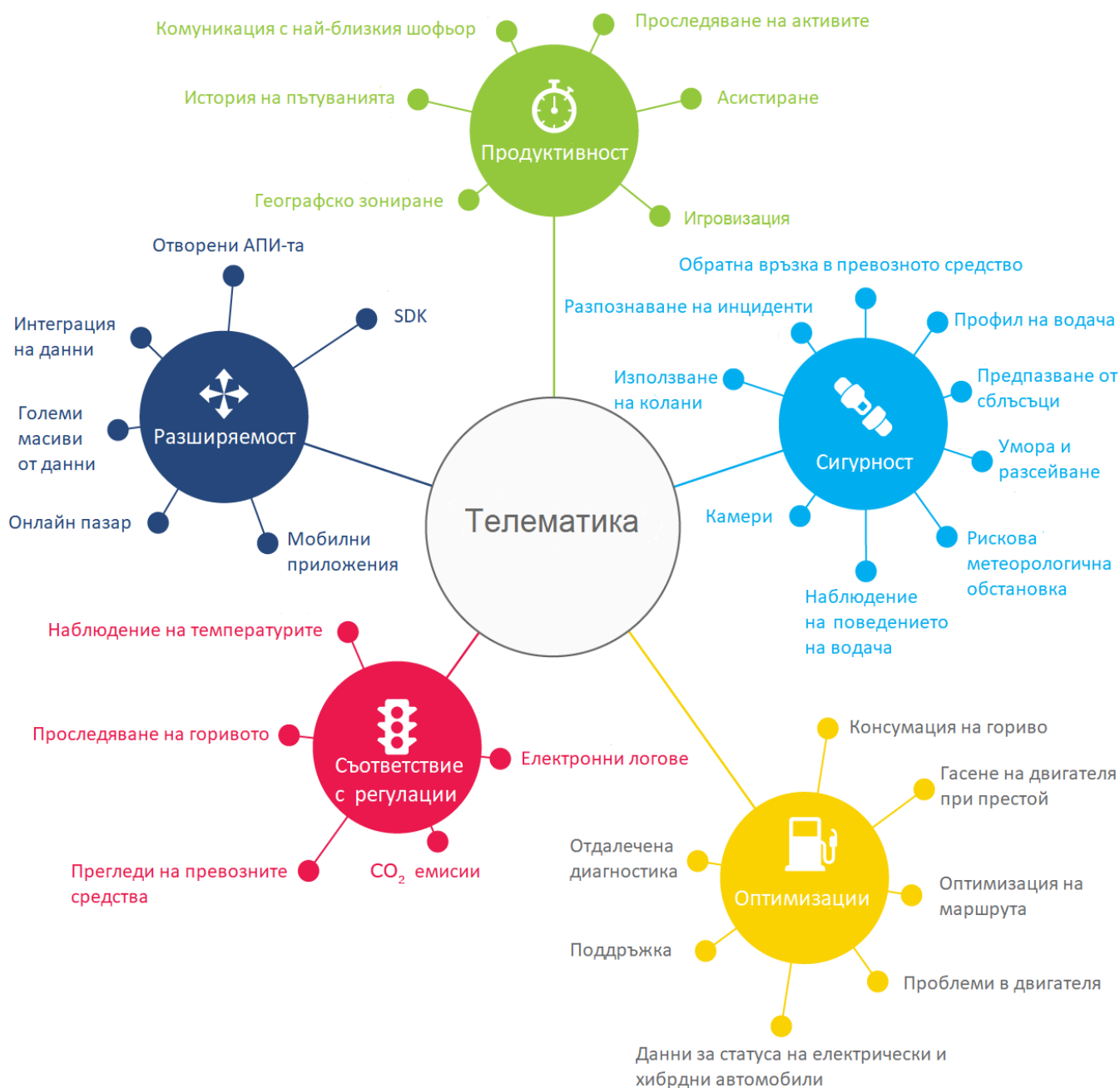
В общия случай използването на устройства от производителите на оригинално оборудване има смисъл, когато собственикът им има доверие и е доволен от

предлаганите от тях телематични услуги. В останалите случаи за предпочитане са страничните производители на телематични устройства.

2.2.6. Приложения на телематиката

На Фигура 4 са показани основните приложения на телематиката. Те са разделени в пет категории:

- **Продуктивност** – Освен чрез по-разпространените механизми като проследяване на активите и асистиране съществуват и някои не толкова познати методи за повишаване на продуктивността. Такава например е игровизацията. При нея игрови елементи се прилагат в ситуации в друг (неигрови) контекст като шофирането. Чрез игрите всеки ден шофьорите се усъвършенстват в спирането, ускоряването и други дейности зад волана, докато споделят постиженията си със своите приятели.
- **Сигурност** – Информацията от автомобилите може да се използва за идентифициране на лоши модели на шофиране, рязко спиране и ускорение. Чрез телематичните устройства могат да се направят предупреждения в реално време за намаляване на скоростта или слагане на предпазен колан. Използването на телематика за прогнозиране на трафика например помага на водачите да избегнат претоварените пътища и потенциални опасности. [19]
- **Оптимизации** – Идентифицирането на лоши модели на шофиране и създаването на добри навици намалява разходите за гориво и поддръжка на автомобила. Освен това възможността за отдалечено проследяване на различни характеристики на превозното средство значително улеснява неговата поддръжка. [19]
- **Съответствие с регулации** – Използване на оптимални маршрути и намаленият разход на гориво допринасят за понижаване на емисиите на въглероден диоксид и спазването на различни регулации. [19]



Фигура 4: „Приложения на телематиката” [7]

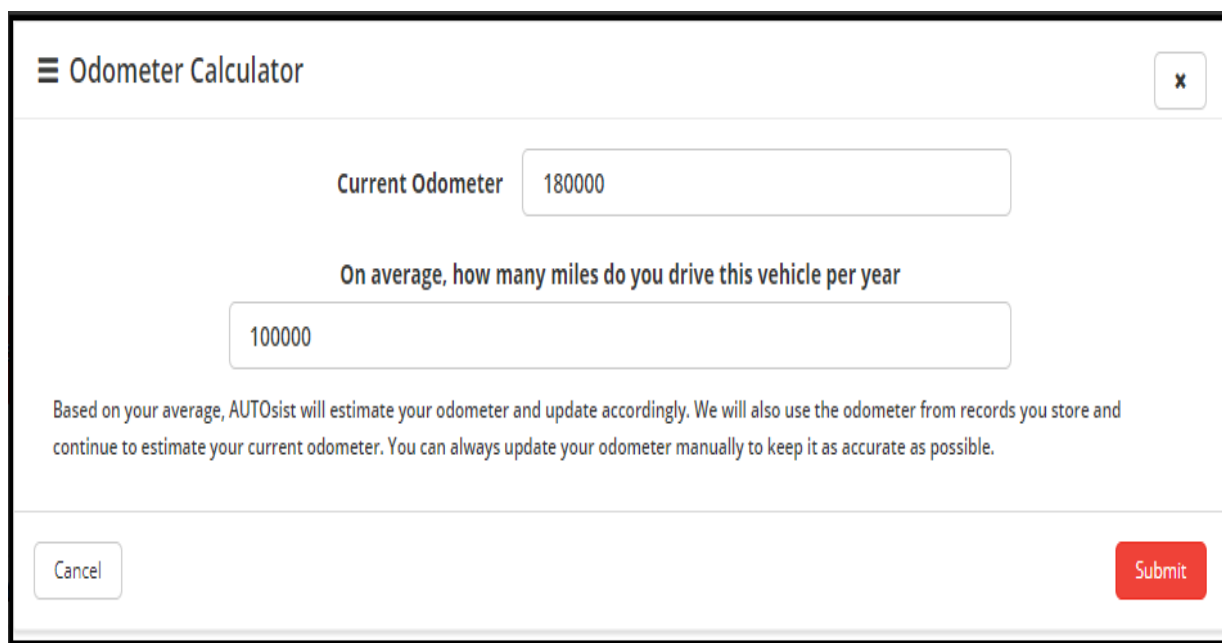
2.3. Съществуващи решения

2.3.1. Autosist

Autosist е популярно приложение за управление на флотилия от превозни средства. То улеснява поддръжката на автомобилите, управлението на гориво и други дейности по управлението на автопарка. Приложението е достъпно като уеб и десктоп версия (за *Windows* и *Mac*), а също и като мобилно приложение за *Android* и *IOS*. [21]

Системата би била най-полезна за бизнес, който не иска да инвестира в закупуването на телематично оборудване. При *Autosist* потребителят сам въвежда текущия километраж

и средния годишен километраж за всяко превозно средство (Фигура 5). Въз основа на тази информация софтуерът изчислява приблизителния километраж към даден момент. За да са по-точни изчисленията е препоръчително регулярно обновяване на текущия километраж на автомобила в системата. Това сякаш е и най-големият недостатък – дори и компанията да притежава автопарк с малко на брой автомобили, за да използва *Autosist*, е необходимо потребителят да въвежда тези данни за всяко превозно средство. Освен това изчисленията никога няма да бъдат с добра точност, защото са базирани на усреднени данни за километража на превозното средство. Напълно възможно е през дадена година превозното средство да се използва по-активно, отколкото предходните и да измине по-голям километраж.



Фигура 5: „Калкулатор за километража” [20]

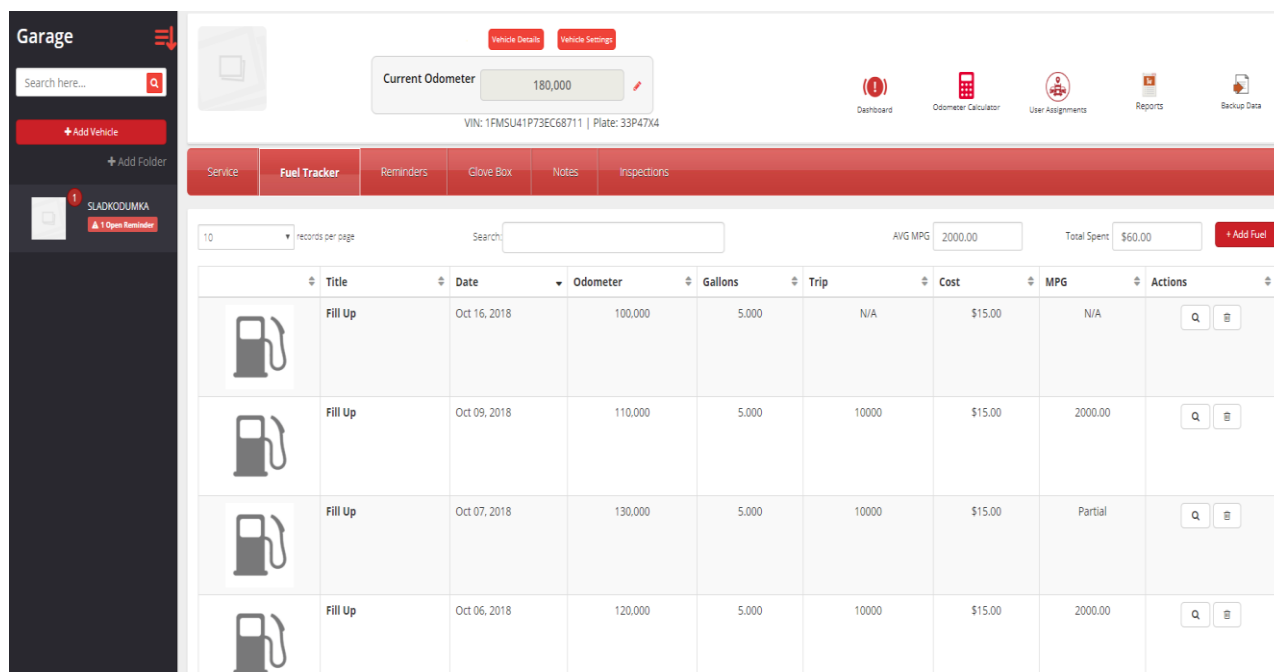
Според някои източници системата е насочена към всички категории компании, включително големи корпорации. Въпреки това те не биха имали голяма полза от подобна система поради ръчното въвеждане на данните за много автомобили и недобрата точност на изчисленията.

Ако пренебрегнем факта, че за да се използват, повечето модули на системата изискват въвеждане на информация от страна на потребителя (поради липсата на телематика), като цяло *Autosist* предлага немалък набор от функционалности (Фигура 6) [20]:

- **Обслужване** – Предоставя функционалности за управление на дейностите по поддръжката на автомобила. Чрез този модул се записват цените на различни части, магазините, от които са закупени, датата на монтиране и други.
- **Проследяване на горивото** – Използва се регистриране на всяко зареждане на автомобила с гориво. При въвеждане на цена за галон и заредено количество, системата изчислява разхода за това зареждане. Възможно е да бъде

пресметнато и изразходваното гориво за дадено разстояние, но за целта потребителят трябва ръчно да въведе началния и крайния километраж.

- **Нотификации** - Системата позволява създаването на нотификации, базирани на време и километраж, които напомнят за извършването на важни дейности.
- **Качване на документи;**
- **Бележки;**
- **Инспекции** – Списъци за проверка на различните части от превозното средство;



Фигура 6: „Основни модули на Autosist” [20]

Моделът на плащане е базиран на броя въведени превозни средства в системата. Предлагат се два плана – основен и така наречения *Fleet* план. Основният пакет е безплатен и той е за лична (некорпоративна) употреба. Той позволява създаването на един потребител и едно превозно средство. *Fleet* планът струва \$50 месечно и позволява регистрирането на повече от пет превозни средства, неограничен брой потребители, устройства и записи, а също и използването на някои функционалности, които не са включени в основния план. Системата предлага тестова версия.

2.3.2. Maintenance PRO WEB [22]

Maintenance PRO WEB е облачно базирана уеб система за управление на дейностите по поддръжката на флотилия от превозни средства, разработена от компанията IMS (*Innovative Maintenance Systems*). Компанията е специализирана в създаването на такъв вид софтуер и предлага на пазара следните решения:

- **Maintenance PRO WEB**
- **Fleet Maintenance Pro** – десктоп приложение за управление на флотилия от превозни средства на колела като автомобили, камиони, автобуси, строителни машини и други;
- **Maintenance Pro** – десктоп приложение за управление на колесни и неколесни превозни средства (самолети, кораби), а също и други машини (например в производствения процес);

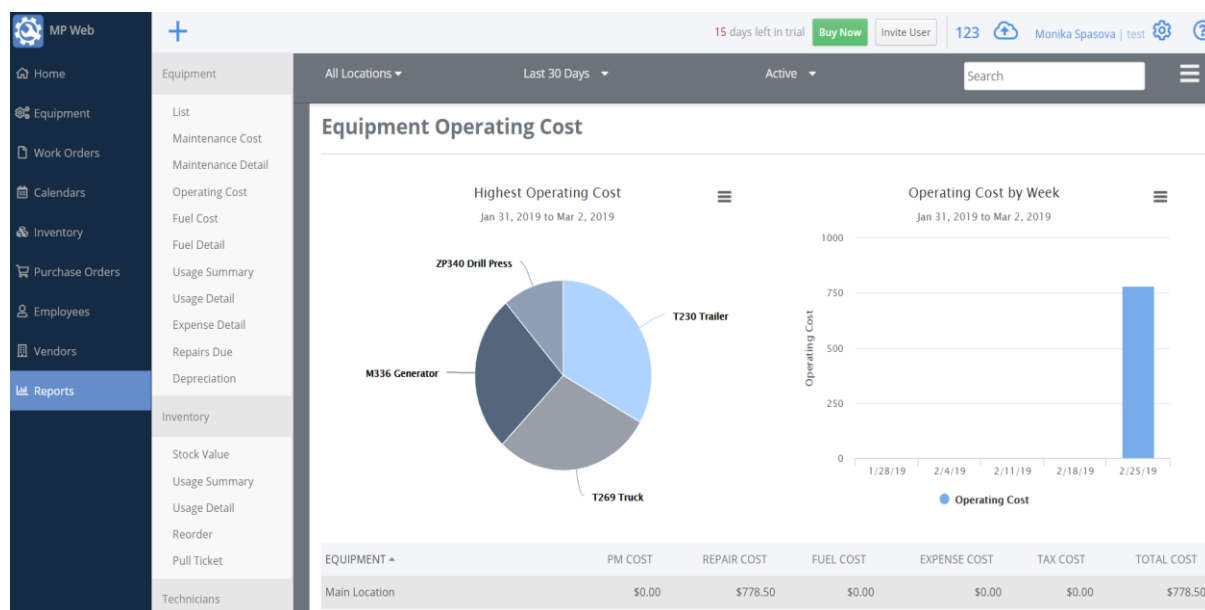
Тъй като софтуерът, разработван в настоящата дипломна работа е уеб, системата *Maintenance PRO WEB* ще бъде разгледана по-подробно. Приложението може да бъде използвано през уеб браузър на устройство с всякакви размери, включително смартфони и таблети. Насочено е към малкия и средния бизнес. Има тестова версия и се таксува чрез месечен абонамент от \$15. Системата предоставя функционалности, които могат да бъдат използвани за управлението на разнообразно оборудване в една компания (не само нейния автопарк). Заради по-широката и специализация не са включени някои функционалности, характерни за управлението на автопаркове. Такива например са използването на километража и нотификациите в реално време.

Подобно на *Autosist* тази система също не е интегрирана с телематично оборудване. Но за разлика от нея тя дори не предлага функционалност за въвеждане на километража от потребителя. Затова управлението на задачите е базирано единствено на посочените от потребителя времеви срокове. Освен това системата не изпраща нотификации по имейл при приближаване на датата за извършване на някаква дейност или при нейното просрочване.

Maintenance PRO WEB предоставя следните функционалности (Фигура 7):

- **Управление на оборудването** – Той позволява въвеждане на превозни средства, създаване на задачи към тях, качване на документи и история на дейностите на превозните средства;
- **Управление на поръчките** – Поръчките се насочват към служителите и имат различен статус, който се обновява в процеса на работа;
- **Календар** – На него се визуализират задачите и поръчките;
- **Управление на инвентара** – Чрез този модул се въвеждат всички активи на компанията. Използва се за управление на техния брой, цена, място на съхранение и други;
- **Управление на покупките** – Използва се за описание на направените покупки, техните купувачи и доставчици, фактури и начин на доставяне на продуктите;
- **Управление на служителите** – Чрез този модул се управлява информацията за служителите, тяхната локация и възнаграждение;
- **Управление на търговците** – Използва се за описание на търговците, с които компанията има бизнес отношения. Съхраняват се техните адреси и информация за контакти.

- **Репорти** – Системата има модул за визуализация на статистика за цените оборудването, неговата поддръжка, статусът на задачите и други за избран от потребителя период;



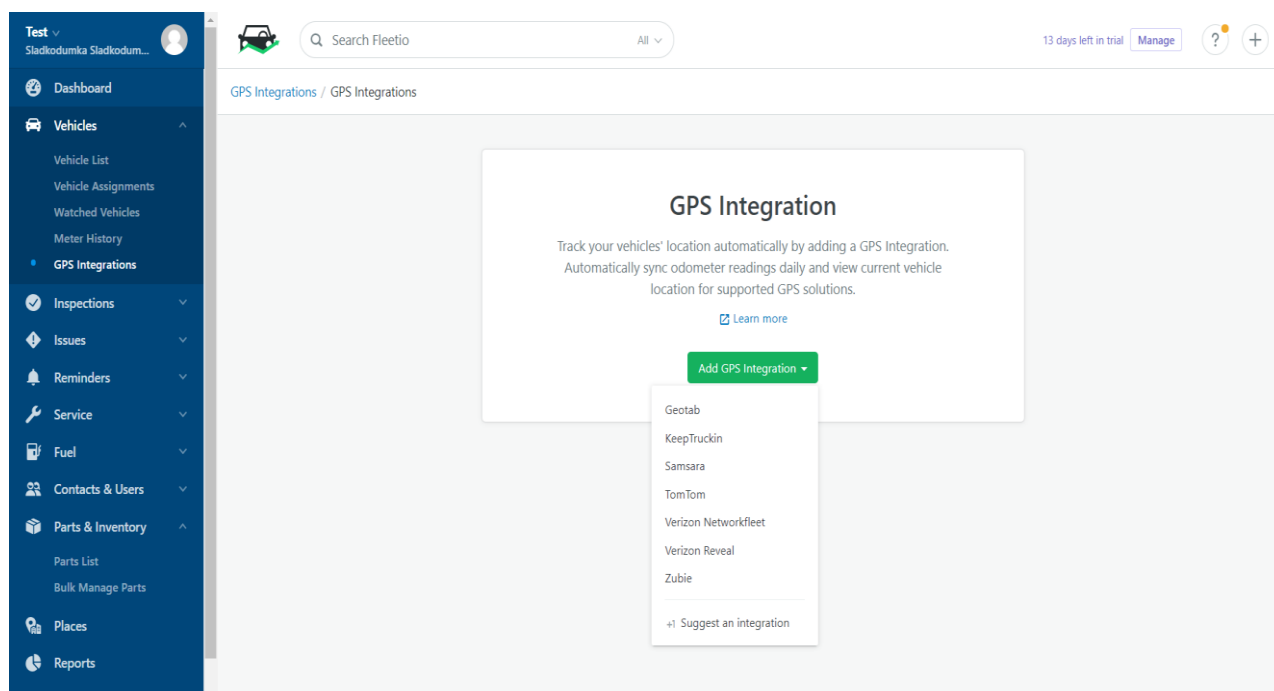
Фигура 7: „Репорти в Maintenance PRO WEB” [22]

2.3.3. FLEETIO [23]

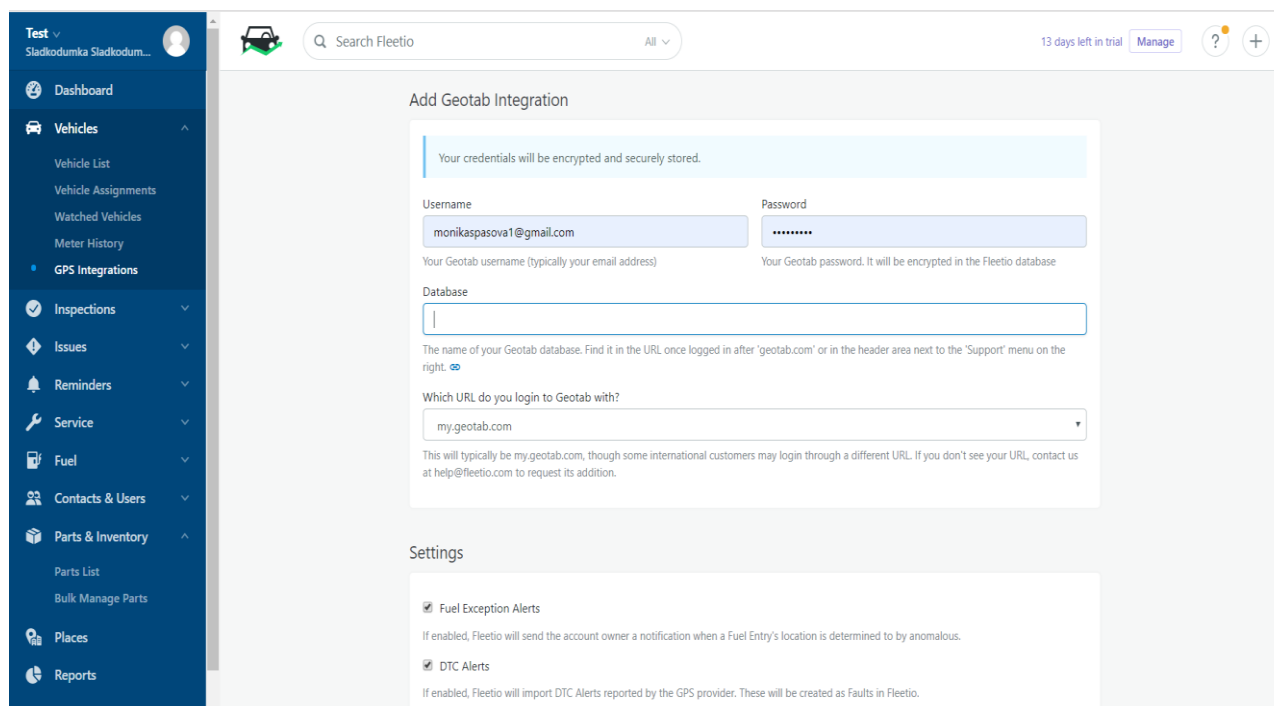
FLEETIO е система за управление на флотилия от превозни средства. Системата има уеб приложение и мобилна версия за *Android* и *IOS*. Използва се от компании с различни по размер автопаркове.

За разлика от *Autosist* и *Maintenance PRO WEB* тази система предлага интеграция с някои доставчици на GPS и телематично оборудване. Чрез опцията „GPS интеграция” в модула „Превозни средства” потребителят може да избере доставчик на оборудване от партньорите на *FLEETIO* (Фигура 8). След това, за да завърши интеграцията, е необходимо въвеждането на името на базата от данни на производителя на оборудването (Фигура 9). Така, ако компанията има автомобили, в които са монтирани устройства на някои от партньорите на *FLEETIO*, тя би могла да използва тази система за управление на своя автопарк. Но ако притежава телематично оборудване от друг производител, различен от партньорите на *FLEETIO*, то интеграцията не би била възможна. Създадената зависимост между софтуер (*FLEETIO*) и хардуер (телематично оборудване) задължава потребителите на *FLEETIO* да се снабдят с оборудване на определен производител, ако искат да използват опцията за интеграция с телематика. Все пак *FLEETIO* се опитва да отговори на търсенето, като предлага анкета, чрез която се допитва до потребителите за техните предпочитания за бъдещи партньори на телематично оборудване. Подобна интеграция би била по-трудна, защото системата не очаква данните от доставчиците в AEM/AEMP формат. Следователно осъществяването

на съвместимост с тях би било далеч по-сложно от техническа гледна точка (различно за всеки доставчик).



Фигура 8: „Възможности за GPS интеграция” [23]



Фигура 9: „Интеграция с доставчик на оборудване” [23]

FLEETIO използва телематичните данни за:

- **Информация за километража** – Данните за километража се обновяват веднъж дневно в системата;
- **За нотификации** – При разпознаване на проблем телематичното устройство изпраща имейл на потребителя. С едно кликане той може да създаде в системата задача за отстраняване на повредата.
- **Проследяване на разходите за гориво** – Системата позволява интеграция с картите за покупка на гориво. Това позволява автоматично следене на разходите за гориво.

В сравнение с *Autosist* и *Maintenance PRO WEB*, *FLEETIO* предоставя по-богати функционалности:

- **Превозни средства** – Позволява въвеждането на превозни средства и подробна информация за тях, включително размери, маса, описание на двигателя, гумите и използваното гориво.
- **Инспекции** – Този модул предоставя списъци за детайлна проверка на всеки от автомобилите. Потребителят може да разгледа резултата от изминали инспекции чрез опцията „История на инспекциите“;
- **Повреди** – Този модул позволява регистрирането на различни повреди на превозните средства в системата;
- **Обслужване на автомобилите** – Към автомобилите могат да се създават различни задачи, свързани с поддръжката им.
- **Напомняне** – Този модул съдържа списък с всички въведени дейности, свързани с обслужването на автомобилите. Чрез различни филтри могат да се видят приближаващите или просрочените задачи.
- **Репорти** – Този модул предлага цялата информация за всяко от превозните средства на едно място чрез списъци и графики.
- **Гориво** – Позволява следене на разходите на горивото, цената на използваното гориво и други;

Приложението има тестова версия. Предлагат се три пакета:

- *Pro* – Включва основни функционалности като управление на гориво, инспекции, повреди, нотификации, търговци и неограничен брой потребители. Цената е \$5 месечно за превозно средство.
- *Advanced* – Този план предоставя някои допълнителни функционалности като управление на инвентара, поръчките и покупките, а също и мобилно приложение. Цената е \$9 месечно за превозно средство.
- *Enterprise* – Включени са възможности за по-високо ниво на поддръжка и функционалности, насочени към потребностите на конкретния бизнес. Цената е по договаряне.

2.3.4. Изводи

След продължително търсене на съществуващи решения, предназначени за малкия и средния бизнес и позволяващи интеграция с телематично оборудване, се оказва, че на практика не съществуват такива. Бяха разгледани три съществуващи решения. За да бъде представено поне едно приложение, което предлага такава интеграция, беше включено *FLEETIO*. Тази система, обаче е насочена предимно към компании с голям автопарк и се предлага на значително по-висок месечен абонамент. Освен това тя не работи съгласно АЕМ/АЕМР стандарта, което значително усложнява интеграцията и с различно оборудване. Тези факти мотивират идеята за разработване на система за управление на флотилия от превозни средства, която е насочена към малкия и средния бизнес и работи съгласно АЕМ/АЕМР стандарта, осигуряващ полесна интеграция с телематичните устройства с различен произход.

3. Използвани технологии

3.1. Изисквания към технологиите

Разработваната система за управление на флотилия от превозни средства е с уеб базирана архитектура. Тя позволява използването и от потребителите чрез уеб браузър. Най-голямото предимство на тази архитектура е нейната разширяемост, тъй като всеки, имащ достъп до системата, може да я използва, откъдето и да е чрез интернет. Като недостатък на уеб приложенията може да се считат потенциални рискове за сигурността и производителността, а също и това, че са неизползваеми при неналична връзка с интернет.

За създаването на уеб системата се използват утвърдени, съвременни уеб технологии. Те са:

- **Зрели** - Съществува достатъчно информация за тяхното използване и решения на потенциални проблеми с тях, а също и възможности за интеграция с разнообразни приложения;
- **Перспективни** – Развиват се и, съдейки по много показатели, ще бъдат използвани в бъдеще. Това улеснява поддръжката на системата в дългосрочен план.

3.2. Използвани технологии [25]

За реализиране на системата са използвани следните технологии:

1. Сървър (*ASP.NET Web API 2*):

- АПИ:
 - **Език** – *Microsoft C#*
 - **Други** – *Microsoft Entity Framework, ASP.NET Identity, Quartz*
- База от данни:
 - **Система за управление на бази данни** – *Microsoft SQL Server*
 - **Език** – *SQL*

2. Клиент (*Alt.js*):

- **Език** – *JavaScript* (стандарт *ECMAScript 6*)
- **Други** – *ReactJS*, *Material UI*, *React-Router*, *Axios.js*, *CSS*

3.2.1. Microsoft Entity Framework [26], [27]

Microsoft Entity Framework е ORM (*Object Relational Mapping*) фреймвурк с отворен код за .NET приложения. Той позволява използването на класове, съответстващи на таблици в базата от данни, вместо самите таблици. Чрез *Entity Framework* е възможна работа на по-високо ниво на абстракция при обработването на данни.

Има три подхода за създаване на класове (модели) чрез *Entity Framework*:

- *Code First*
- *Database First*
- *Model First*

За разработването на системата за управление на флотилия от превозни средства е използван подходът *Code First*. При него програмистът първо създава C# класове. След изпълнението на командата *Update-Database* в конзолата на *Visual Studio* въз основа на тях се генерират SQL таблици. При този подход, за да се модифицира схемата на базата, трябва промяната да се направи в C# модела и да се изпълни командата за обновяване на базата от данни.

3.2.2. ASP.NET Web API 2 [28]

ASP.Net Web API 2 е фреймвурк с отворен код за създаване на уеб приложения, базиран на езика за програмиране *Microsoft C#*. Той осъществява комуникация с клиента чрез HTTP заявки на езика, указан в полето *Content-Type* на HTTP протокола. В разработваната система за управление на флотилия от превозни средства това е JSON.

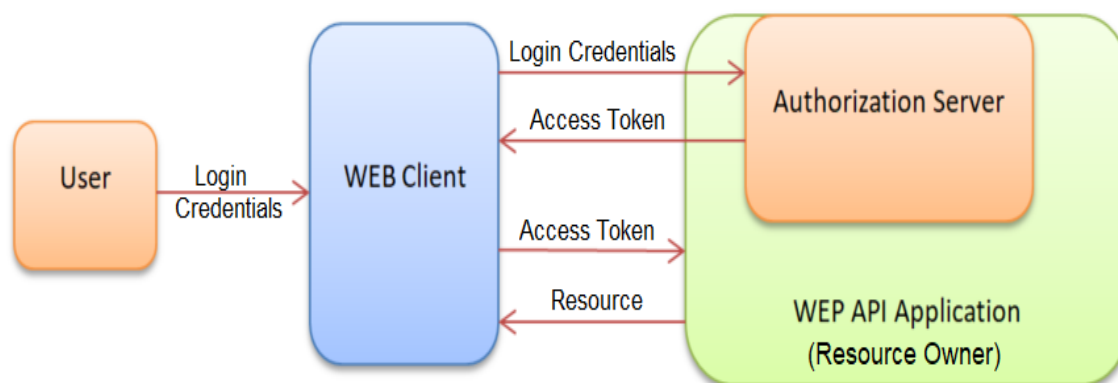
ASP.Net Web API е изграден въз основа на *ASP.NET MVC* и съдържа някои от неговите характеристики като маршрутизация, контролери, модели и други. Освен това този фреймвурк надгражда предшественика си с нови функционалности като *OData* филтрация и възможността да бъде използван от различни клиенти като мобилни приложения. *ASP.Net Web API* може да приема и връща резултат, който не е обектно-ориентиран, например изображения и PDF файлове.

3.2.3. ASP.NET Identity [29], [30]

Защитата в *WEB API 2* може да се осъществи по два начина:

- Основна аутентикация
- Аутентикация с токен

При разработването на функционалността за регистрация и вписване на потребител в системата за управление на флотилия от превозни средства чрез библиотеката *ASP.NET Identity* се използва аутентикация с токен (Фигура 10). При нея клиентът първо изпраща заявка към сървъра за аутентикация с валидни идентификационни данни (потребителско име и парола). След успешна валидация сървърът изпраща токен (извършено е успешно вписване), който клиентът използва за достъп. Този токен е специфичен за всеки потребител и има валидност във времето. Клиентът го съхранява в браузъра и го използва за заявки към *WEB API* приложението. Когато изтече, клиентът прави заявка за нов токен за достъп. В потребителския интерфейс това става чрез пренасочване към страницата за вписване на потребителя.



Фигура 10: „Аутентикация с токен”, [30]

3.2.4. Quartz [31]

Quartz е библиотека за изпълнение на планирани задачи според предварително указани време, интервали и други. В системата за управление на флотилия от превозни средства *Quartz* задачи се използват за:

- *SendMileageReminderEmailJob*, *SendTimeReminderEmailJob* – За изпращането на имейлите за предстоящи дейности по обслужването на автомобила, базирани на време и километраж;
- *SendMileageOverdueEmailJob*, *SendTimeOverdueEmailJob* – За изпращането на имейлите за пресрочени дейности по обслужването на автомобила, базирани на време и километраж;
- *SeedTelematicsJob* – За генерирането на данни в таблиците *TelematicsData* и *TelematicsDataHistory*;

Quartz има следните предимства:

- **Гъвкавост** – Предоставя разнообразни подходи, които могат да бъдат използвани заедно или поотделно;
- **Леснота** – Изисква много малко конфигурации;
- **Устойчива при проблеми** – Продължава да работи дори при проблеми и рестартиране на системите;

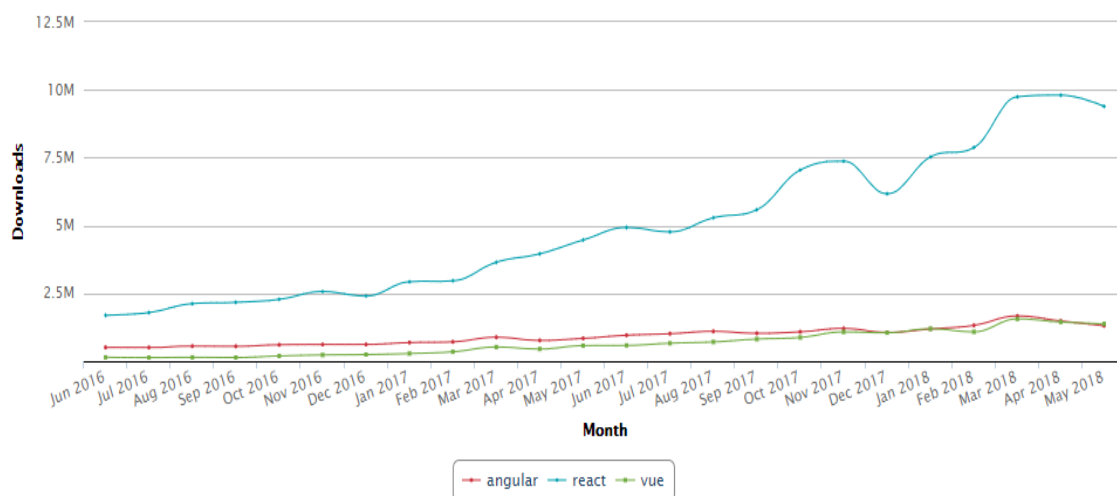
3.2.5. Alt.js

Alt.js е библиотеката, използвана за реализацията на клиентската част на уеб системата. Тя е едно от популярните технологични представяния на идеята *Flux*. Библиотеката осигурява необходимите функционалности и синтаксис, чрез които в кода се извършва комуникацията на основните *Flux* елементи (*Store*, *Action*, *View*). За *View* елемента се използва *ReactJS 16.4.1*.

3.2.6. ReactJS [32], [33], [34]

ReactJS е *JavaScript* библиотеката, с която е разработена визуалната част на уеб приложението. Създадена е през 2011г от *Facebook*, а през 2013г става с отворен код. *ReactJS* трупа голяма популярност за кратко време благодарение на своите особености (Фигура 11):

- **Компоненти** – *ReactJS* въвежда създаването на компоненти. Те могат да бъдат използвани на различни страници в потребителския интерфейс (преизползваеми са).
- **Виртуален DOM (*Document Object Model*)** - Благодарение на виртуалния DOM е възможно създаването на бързи и мащабируеми приложения. Чрез алгоритъма *Memory reconciliation* библиотеката представя страниците във виртуалната памет. Там се извършват необходимите промени преди крайното визуализиране на страницата в браузъра. [30]
- **Лесен за научаване** - *ReactJS* е много по-лесна за научаване от други *JavaScript* библиотеки поради простотата по отношение на синтаксиса.
- **Голяма степен на гъвкавост;**
- **Високо бързодействие** – При работа с *EcmaScript 6* и *EcmaScript 7*, *ReactJS* работи бързо при голямо натоварване;
- **Лесна миграция между различните версии; [31]**



Фигура 11: „Статистика на броя сваляния от GitHub на най-популярните JavaScript библиотеки” [34]

3.2.7. React-Router

React-Router е библиотека, която позволява навигацията между различните страници на *ReactJS* приложение. За реализацията на системата е използван *React-Router 3.0.5*.

3.2.8. Axios.js

Axios.js се използва за извършването на HTTP заявки от клиента към сървър. В разработваното приложение е използвана *Axios 0.18.0*.

3.2.9. Material-UI [35]

Material-UI е библиотека с отворен код, лицензирана от MIT. Тя предлага голямо разнообразие от *ReactJS* компоненти по *Google's Material Design*. Библиотеката е разработена през 2014г и се превръща в една от най-използваните за потребителския интерфейс на приложения, използващи *ReactJS*.

Основните и предимства са:

- **Лесно използваема** – Библиотеката предлага богата документация и много примерни проекти, онагледяващи използването на различните компоненти;
- **Изоляция** – Компонентите са така разработени, че при използването им техните стилове не се пренаписват от глобални стилове в приложението;
- **DOM** – *Material-UI* се стреми да създава компоненти на ниско ниво. Тези компоненти са максимално близо до DOM структурата и могат да бъдат лесно променяни според изискванията за дизайна.
- **Качество на кода** – Компонентите са напълно достъпни и съвместими с HTML 5. 100% от кода на библиотеката е покрит от тестове.

Material-UI (версия 3.3.2) компоненти са използвани за почти целия потребителски интерфейс на разработваната система за управление на флотилия от превозни средства. Голямото разнообразие от компоненти (бутони, текстови полета, менюта, таблици, икони и други) с консистентен дизайн допринасят за създаването на пълноценно потребителско изживяване при работа с приложението.

3.2.10. Recharts [36]

За визуализиране на репортите относно характеристиките на превозни средства се използва библиотеката *Recharts*. Тя е разработена въз основа на *ReactJS* и *D3* (една от най-известните библиотеки за диаграми). В разработваната система е използвана графиката *AreaChart*. В настоящата версия на приложението оста X винаги съдържа времето, а Y друг вид данни.

4. Анализ

4.1. Концептуален модел

За осигуряване на достъп до системата първо е необходима регистрация (Фигура 12). След въвеждането на уникално потребителско име, уникален имейл и парола във формата за регистрация потребителят е пренасочен към страницата за вписване. Там чрез попълване на валидно потребителско име и парола се осигурява неговият достъп до началната страница на системата (Фигура 13).

Тя, за разлика от страниците за регистрация и вписване, разполага с главно меню. Всъщност по това дизайнът на тези две страници се различава от останалите страници в системата. При екраните за регистрация и вписване на мястото на главното меню са разположени навигационните бутони *Login* и *Register* (Фигура 12).

Главното меню е достъпно от всяка страница на приложението след успешно вписване. То осигурява бутон за излизане от системата *Logout*, а също и възможност за преглед на основна информация за вписания потребител (Фигура 13).

Освен главно меню началната страница визуализира списък с информацията за компаниите в системата, до които потребителят има достъп. В най-честия случай тя ще бъде само една. Може да са няколко, ако потребителят се занимава с разнообразен бизнес и иска да използва системата за управление на превозните средства за няколко фирми или пък да раздели превозните средства в компанията на няколко групи по дадена категория (например спрямо местоположението на офиса, който обслужват). Бутонът *Create Company* и иконите за редактиране и изтриване позволяват управление на информацията за компаниите. Икона за изтриване се визуализира единствено, ако вписаният потребител е създател (администратор) на тази компания (Фигура 13). Потребителят може да даде достъп до компанията на други регистрирани потребители при (Фигура 14):

- създаването на компанията
- редактиране на създадена от него компания

LOGIN

REGISTER

Username*

Enter your username

Email*

Enter your email

Password*

Enter your password

Confirm password*

Confirm your password

REGISTER

Фигура 12: „Страница за регистрация”

Fleet Management							LOGOUT
CREATE COMPANY							
Name	Address	Email	Telephone	Preview	Edit	Delete	
Super Trans	Madison, 1820 Irish Lane	supertrans123@yahoo.com	0414261200				
Lux Transport	Jackson, 2340 Tanglewood Road	luxTransport@gmail.com	0822405422				
Taxi ABC	Houston, 2546 Wines Lane	taxi_abc@outlook.com	0866959555				
Super Trucks	San Francisco, 4141 Duck Creek Road	super_trucks@outlook.com	0875253952				

Фигура 13: „Начална страница”

Fleet Management

LOGOUT

COMPANIES

Name*

Super Trans

Email*

supertrans123@yahoo.com

Address*

Madison, 1820 Irish Lane

Phone

0414261200

Allow access to users

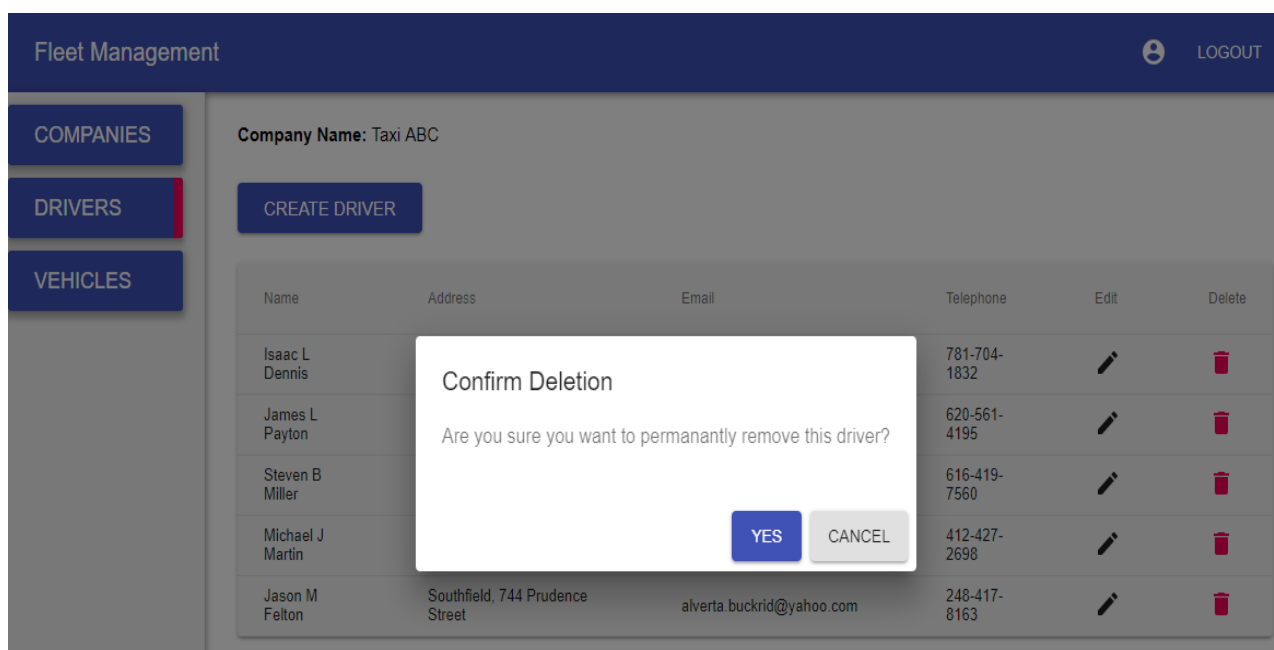
Evelina

George

SAVE

Фигура 14: „ Редактиране на компания ”

За да се възползва от останалите функционалности на системата потребителят трябва да кликне върху иконата *Preview* срещу избраната от него компания от началната страница (Фигура 13). Тогава той бива пренасочен към модула за управление на шофьорите към дадената компания. На екрана се появява и странично меню за навигация между отделните модули на системата. При натискане на бутона *Companies* потребителят може да се върне към списъка с компаниите. Ако все пак реши да остане на страницата за преглед на шофьорите, той може да се насочи към създаване на нов шофьор в системата, редактиране или изтриване на съществуващ. При натискане на иконата за изтриване на някой шофьор се визуализира прозорец за потвърждение на операцията (Фигура 15). Такъв се показва и при изтриването на записи в останалите модули на системата.



Фигура 15: „Изтриване на шофьор”

Потребителят достъпва модула за превозни средства чрез бутона *Vehicles* от страничното меню. Така той може да прегледа съществуващите превозни средства и да извършва CRUD (*Create, Upadte, Delete*) операции с тях. При създаване или редактиране на превозно средство е възможно избирането на съществуващ в системата шофьор, който да бъде свързан с превозното средство (Фигура 16). Задължителни за създаване на превозно средство в системата са полетата VIN, регистрационен номер, тип (например лек автомобил или камион) и марка.

Fleet Management

COMPANIES

DRIVERS

VEHICLES

Company Name: Taxi ABC

VIN*

2G1WW12E939453902

Plate Number*

142-QDE

Type*

car

Brand*

Renault

Model

Clio

Isaac L Dennis

James L Payton

Steven B Miller

Michael J Martin

Jason M Felton

Фигура 16: „Създаване на превозно средство”

На страницата за преглед на превозни средства при избор на иконата *Preview* на някой ред от списъка потребителят е пренасочен към страницата за управление на дейностите (задачите) към избрания автомобил (Фигура 17). Тя се различава по някои UI компоненти от страниците за преглед в останалите модули. При нея под информацията за компанията и превозното средство е разположено още едно меню с два бутона *Services* и *Reports*. То се използва за навигация между двата модула. Освен това до бутона за създаване на задача е разположен бутон за филтрация на пресрочените дейности. В списъка с дейности има една допълнителна икона *Mark as done* за отбелязване на задача като завършена.

Fleet Management

LOGOUT

COMPANIES

DRIVERS

VEHICLES

Company Name: Taxi ABC

Brand: Toyota

Plate Number: 136-SMTR

SERVICES

REPORTS

CREATE SERVICE

☐ Overdue services

Name	Description	Mileage Rule	Next Service Mileage	Mileage Reminder	Next Service Mileage Reminder	Time Rule	Next Service Time	Time Reminder	Next Service Time Reminder	Mark as done	Edit	Delete
Car Wash						1 m	27/02/19	2 d	25/02/19			
Change the brake fluid		5000km	5343km	300km	5043km							
Year preventative maintenance						1 y	27/01/20	1 m	27/12/19			
Fuel Change						1 d	28/01/19	1 d	27/01/19			
Change the oil		5000km	5343km	300km	5043km							

Фигура 17: „Преглед на дейности по обслужването на превозно средство”

Във формата за създаване на задача потребителят освен име и реципиент на нотификациите трябва задължително да попълни полетата към една от двете категории – дейност основана на време или километраж (Фигура 18).

Ако задачата е базирана на времето, потребителят трябва да въведе:

- времето, през което иска да се изпълнява задачата, в полетата *Time Rule* и *Time Rule Unit*;
- колко време преди настъпването на датата за изпълнение на задачата иска да получи напомняне в полетата *Time Reminder* и *Time Reminder Unit*;

При създаване на задачата въз основа на въведените данни и спрямо днешната дата се изчисляват датите за следващото напомняне и извършване на насрочената дейност. Те се визуализират в колоните *Next Service Time* и *Next Service Time Reminder* в списъка със задачите (Фигура 17).

В случай че задачата е основана на километража, потребителят трябва да въведе :

- километража, през който иска да се изпълнява задачата, в полето *Mileage Rule*;
- колко километра преди достигането на километража за изпълнение на задачата, иска да получи напомняне в полето *Mileage Reminder*;

При натискане на бутона за запазване на задача, базирана на километража, спрямо текущия километраж се изчисляват километражът, на който потребителят ще получи

напомняне, а също и километражът за изпълнение на задачата. Те се визуализират в списъка със задачи в колоните *Next Service Mileage* и *Next Service Mileage Reminder* (Фигура 17).

Fleet Management

COMPANIES

DRIVERS

VEHICLES

Company Name: Taxi ABC

Brand: Toyota

Plate Number: 136-SMTR

SERVICES

REPORTS

Name*

Change the tires

Notifications recipient*

monikaspasova1@gmail.com

Description

Mount the winter tires

Select time-based or mileage-based notifications*

☒ Time ☐ Mileage

Time Rule*

1

Time Rule Unit

Years

Time Reminder*

14

Time Reminder Unit

Days

SAVE

Фигура 18: „Създаване на задача към превозно средство, основана на времето”

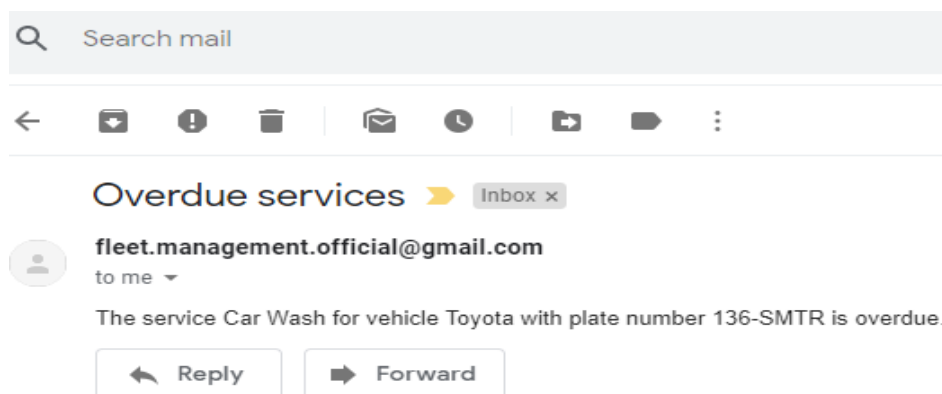
Веднъж дневно системата изпраща имейли към реципиентите за предстоящи и пресрочени дейности. Изпращането на имейлите се основава на:

- сравнение на настоящата дата с датите *Next Service Time* и *Next Service Time Reminder* от списъка (Фигура 17);
- сравнение на настоящия километраж със стойностите *Next Service Mileage* и *Next Service Mileage Reminder* от списъка (Фигура 17). Тъй като изпращането на имейлите се извършва веднъж дневно, почти невъзможно е в момента на проверката стойностите на *Next Service Mileage Reminder* и текущия километраж на превозното средство да са равни. Затова за изпращането на напомняне за задача, основана на километража, се проверява дали $NextServiceReminderMileage \in [vehicleTelem.Mileage-50, vehicleTelem.Mileage+50]$.

Изпращат се четири вида имейли:

- за предстояща задача, базирана на време
- за пресрочена задача, базирана на време
- за предстояща задача, базирана на километраж
- за пресрочена задача, базирана на километраж

Те изглеждат по сходен начин. Заглавието е *Reminder services* или *Overdue services*. В съдържанието на имейла се споменават името на задачата, марката превозно средство и неговия регистрационен номер (Фигура 19).



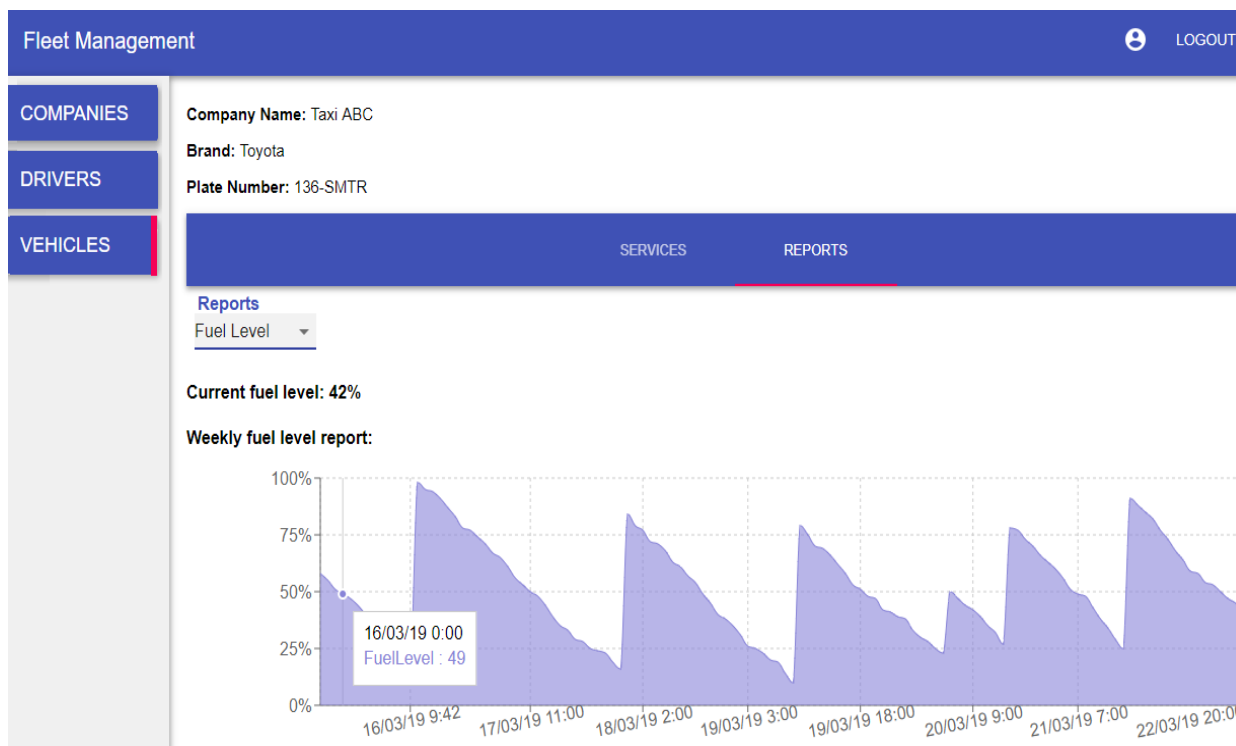
Фигура 19: „Имейл за пресрочена дейност”

Потребителят отбелязва задача като завършена чрез кликване върху иконата *Mark as done* от списъка със задачите (Фигура 17). При потвърждение на операцията:

- При задача, основана на времето, спрямо днешната дата и въведените правила се изчисляват следващата дата за напомняне и изпълнение на задачата;
- При задача, основана на километража, спрямо текущия километраж и въведените правила се изчисляват следващия километраж за напомняне и изпълнение на задачата;

При избиране на бутона *Reports* потребителят е пренасочен към модула с репорти за превозното средство. От падащия списък *Reports* потребителят може да избере две опции (Фигура 20):

- *Mileage* – При тази опция на екрана се визуализират текущия километраж на автомобила и чрез графика статистика за стойностите на километража за период от една седмица назад.
- *Fuel* – Показва се текущото ниво на горивото на превозното средство в проценти и статистика за същото за една седмица назад.



Фигура 20: „Репорт на горивото на превозно средство”

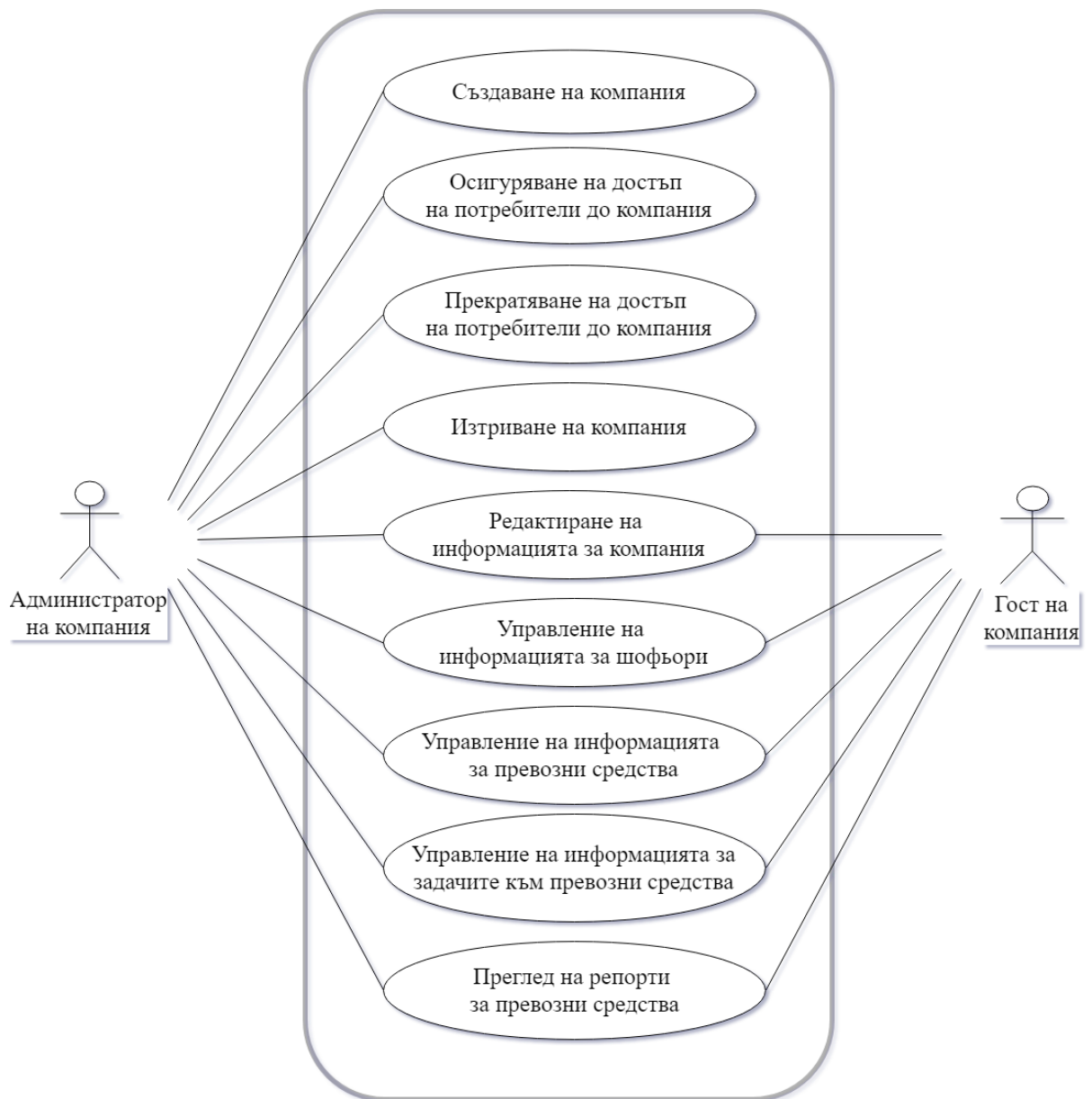
4.2. Функционални изисквания

Основните роли според достъпа до системата са:

- **Регистриран потребител** – Потребителят е направил регистрация в системата.
- **Вписан потребител** – Потребителят е направил регистрация в системата и чрез формата за вписване е влязъл в нея.
- **Нерегистриран потребител**

Основните роли в системата според достъпа до компания са (Фигура 21):

- **Администратор на компания** – Тази роля се изпълнява от създателя на компания. Той има право да управлява достъпа на регистрирани потребители до администрираната от него компания, а също и да я изтрива.
- **Гост на компания** – Това е регистриран потребител, на когото администраторът е дал достъп до дадена компания. Той, обаче няма право да променя достъпа на други потребители до нея и да я изтрива. По отношение на останалите функционалности правата на администратора и госта на компания не се различават.
- **Няма достъп до компания** – Регистриран потребител, на когото администраторът не е дал достъп до компанията.



Фигура 21: „Use-case диаграма на системата”

Дефинирани са следните функционални изисквания:

- 1) Системата да разполага с регистрационна и логин форма.
- 2) Системата да се администрира чрез:
 - главно меню
 - странично меню
- 3) Системата да предоставя възможност на вписания потребител да:
 - създава компания
 - осигурява и прекратява достъп на регистрирани потребители до създадената от него компания
 - редактира информацията на създадена от него компания или на компания, до която има достъп

- изтрива, създадена от него компания
 - преглежда списък с информацията на създадените компании
- 4) Системата да осигурява възможност на администратор и гост на компанията да:
- създават профил на шофьор
 - редактират информацията за шофьора
 - изтриват профила на шофьора
 - преглеждат списък с информацията на въведените шофьори
- 5) Системата да осигурява възможност на администратор и гост на компанията да:
- създават превозно средство
 - редактират информацията за превозно средство
 - изтриват превозно средство
 - свързват превозно средство с шофьор от компанията
 - преглеждат списък с информацията на въведените превозни средства
- 6) Системата да осигурява възможност на администратор и гост на компанията към превозно средство да:
- въвеждат задача, базирана на времето
 - въвеждат задача, базирана на километража
 - въвеждат имейл адрес, на който да получава нотификации за предстоящи и пресрочени задачи
 - редактират задачи
 - изтриват задачи
 - преглеждат списъци с задачи
 - получават нотификации за предстоящи и пресрочени задачи
 - филтрират пресрочените задачи
 - преглеждат информация за километража и нивото на горивото в реално време
 - преглеждат информация за километража и нивото на горивото за една седмица назад чрез графика

4.3. Нефункционални изисквания

- Системата да бъде съвместимата браузърите *Google Chrome*, *Microsoft Edge*, *Mozilla Firefox* и *Opera*.
- Потребителският интерфейс на системата да бъде на английски език.
- Всяко взаимодействие на потребителя със системата не трябва да отнема повече от 3 секунди.
- Да няма специфични изисквания към техническите характеристики на машината, на която системата ще бъде използвана.

4.4. Качествени атрибути

4.4.1. Използваемост

Системата има за цел да бъде разбираема и достъпна за потребителите. Архитектурата на потребителския интерфейс трябва да предоставя на потребителя удобство при работа с приложението:

- Бутони, линкове, икони, списъци, форми за въвеждане на данни - Да се подчиняват на цялостен дизайн и да не са в противоречие с утвърдените интерактивни модели. Всички текстови полета трябва да бъдат озаглавени и да съдържат съобщение, указващо каква да бъде въведената информация. Задължителните полета да бъдат означени със звезда.
- Валидации на входните данни;
- Прозорци за потвърждение при изтриване и маркиране на дейност като завършена;

4.4.2. Сигурност

Системата да предоставя сигурен достъп:

- Оторизиран достъп – От всеки потребител, желаещ да използва приложението, се изисква регистрация. Системата трябва да разполага със защита срещу неоторизиран достъп, като ограничава възможността за достигане до дадена страница само с написването на адреса на страницата в полето на брауъра. Ако нерегистриран потребител или потребител с изтекъл токен за удостоверяване се опита да достъпи дадена страница от системата, той трябва да бъде пренасочен към страницата за регистрация/вписване.
- Валидация на въведените потребителски данни;
- Хеширане на паролите при съхранението им в базата от данни;
- Сложни пароли – Валидните потребителски пароли е необходимо да съдържат букви, цифри и специални символи;

Всичко това допринася за добрата защита на системата от кибер атаки.

4.4.3. Модифицируемост

Системата трябва да бъде лесно модифицируема. За целта се налага използването на компоненти на ниско ниво и отделянето на преизползваемата логика във функции. Необходимо е използването на REST архитектура, която също допринася за модифицируемостта на системата. Така например клиентът може да бъде сменен с такъв, който е базиран на други технологии, и това няма да повлияе по никакъв начин на сървърната част и обратното.

4.4.4. Мащабируемост

Всяко ниво на системата трябва да може лесно да се разширява без да пречи на останалите.

4.4.5. Поддръжка

Системата да бъде лесна за поддръжане. Заради това е необходимо тя да има модулна архитектура и ясно разделение на функциите на различните нива.

4.5. Работни процеси

По-подробно ще бъдат разгледани някои от основните работни процеси на потребителя със системата:

4.5.1. Регистрация

Този процес се извършва от нерегистрирани потребители. Състои се в следното:

1. Отваряне на системата в браузъра;
2. Натискане на бутона *Register* от навигационната лента;
3. Попълване на регистрационната форма с потребителско име, имейл, парола и потвърждение на паролата;
4. Натискане на бутона *Register*;
5. При въвеждане на валидни данни (уникално потребителско име, валиден имейл, парола и правилно потвърждение) преминаване към 6, иначе към 3;
6. Пренасочване към страницата за вписване;

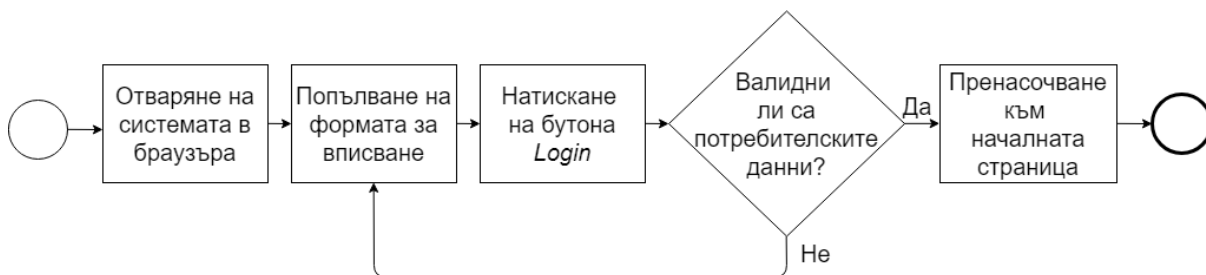


Фигура 22: „Процес по регистрация в системата”

4.5.2. Вписване

Процесът се извършва от регистрираните потребители. Включва следните стъпки:

1. Отваряне на системата в браузъра;
2. Попълване на формата за вписване с потребителско име и парола;
3. Натискане на бутона *Login*;
4. При въвеждане на валидно потребителско име и парола преминаване към 5, иначе към 2;
5. Пренасочване към началната страница на системата;

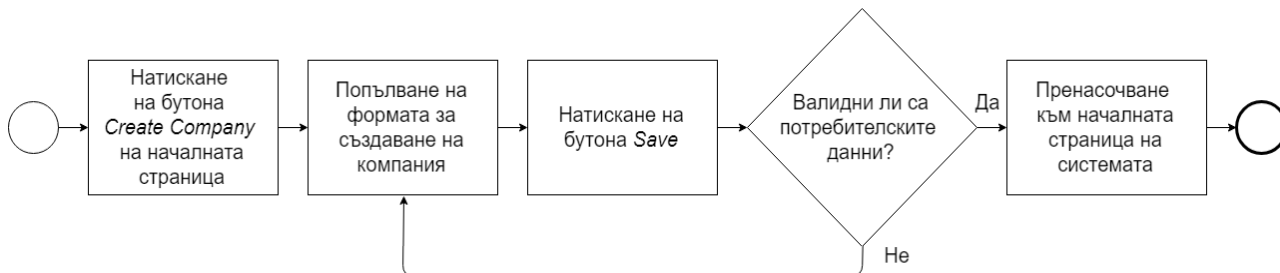


Фигура 23: „Процес по вписване в системата”

4.5.3. Създаване на компания

Процесът се извършва от вписан потребител и се изразява в следните стъпки:

1. Натискане на бутона *Create Company* от началната страница на системата;
2. Попълване на формата за създаване на компания със задължителните полета (име, имейл, адрес) и по избор опционалните (телефонен номер и осигуряване на достъп на регистрирани потребители);
3. Натискане на бутона *Save*;
4. При успешна валидация на входните данни преминаване към 5, иначе към 2;
5. Пренасочване към началната страница на системата;



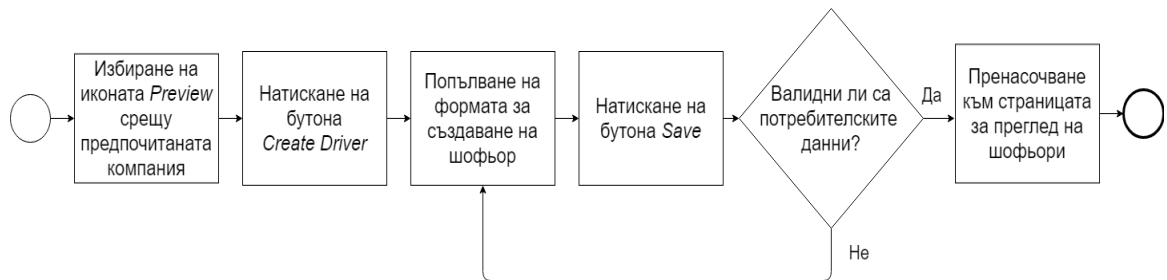
Фигура 24: „Процес по създаване на компания”

4.5.4. Създаване на шофьор към компания

Този процес се извършва от администратор на компания или гост на компания, вписани в системата. Той включва следните стъпки:

1. Избиране на иконата *Preview* срещу компания от списъка с компании на началната страница на системата;
2. Натискане на бутона *Create Driver*;
3. Попълване на формата за създаване на шофьор със задължителните полета (име, имейл, адрес) и по избор телефон;
4. Натискане на бутона *Save*;

5. При успешна валидация на входните данни преминаване към 6, иначе към 3;
6. Пренасочване към страницата за преглед на шофьори, където в списъка е добавен новосъздадения шофьор;

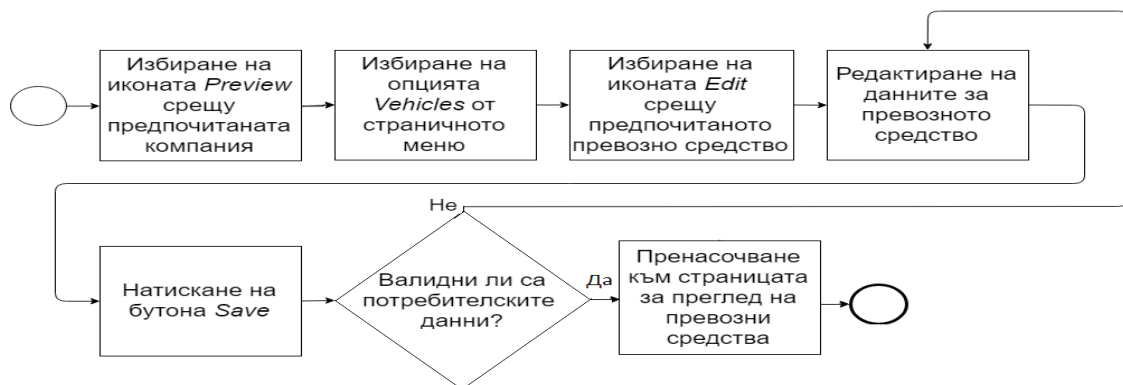


Фигура 25: „Процес по създаване на шофьор към компания”

4.5.5. Редактиране на превозно средство към компания

Извършва се от администратор на компания или гост на компания, вписани в системата. Изразява се в следните стъпки:

1. Избиране на иконата *Preview* срещу компания от списъка с компании от началната страница на системата;
2. Избиране на опцията *Vehicles* от страничното меню;
3. Избиране на иконата *Edit* срещу превозното средство от списъка, което потребителят иска да редактира;
4. Редактиране на избрани полета от формата за промяна на превозно средство;
5. Натискане на бутона *Save*;
6. При успешна валидация на входните данни (непразни задължителни полета – VIN, регистрационен номер, вид и марка) преминаване към 7, иначе към 4;
7. Пренасочване към страницата за преглед на превозни средства, където в списъка е обновена информацията за редактираното превозно средство;

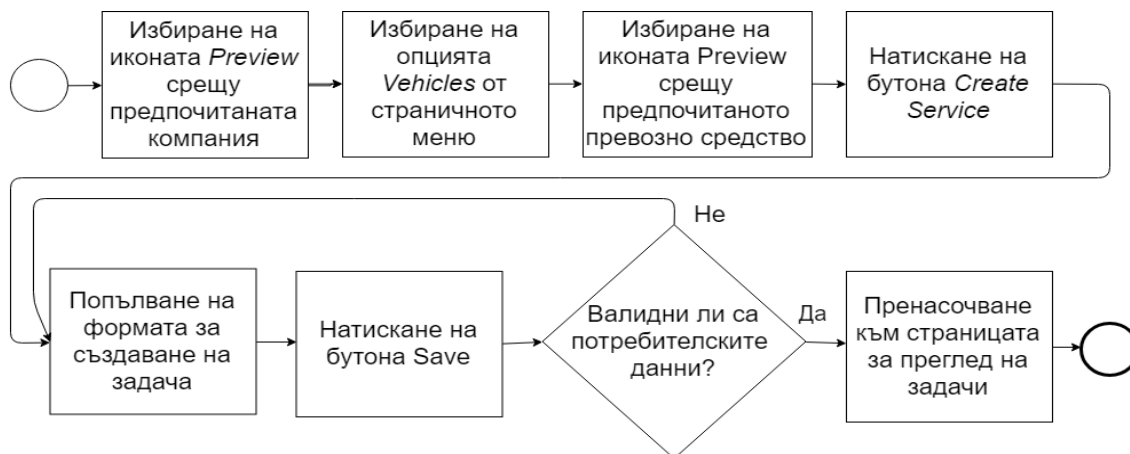


Фигура 26: „Процес по редактиране на превозно средство”

4.5.6. Създаване на задача към превозно средство

Процесът се извършва от администратор на компания или гост на компания, вписани в системата. Той се изразява в следните стъпки:

1. Избиране на иконата *Preview* срещу компания от списъка с компании от началната страница на системата;
2. Избиране на опцията *Vehicles* от страничното меню;
3. Избиране на иконата *Preview* срещу превозното средство от списъка, към което потребителят иска да добави задача;
4. Пренасочване към страницата за преглед на задачи;
5. Натискане на бутона *Create Service*;
6. Попълване на задължителните данни:
 - а. Име и реципиент за нотификации;
 - б. Базиране на задачата на време или километраж;
 - в. Правилата за извършване на дейност и нотификации въз основа на избраната опция (време или километраж);
7. Натискане на бутона *Save*;
8. При успешна валидация на входните данни преминаване към 9, иначе преминаване към 6;
9. Задачата е добавена успешно, преминаване към 4;



Фигура 27: „Процес по създаване на задача към превозно средство”

4.5.7. Филтрация на пресрочени задачи към превозно средство

Процесът се извършва от администратор на компания или гост на компания, вписани в системата. Той включва следните стъпки:

1. Избиране на иконата *Preview* срещу компания от списъка с компании от началната страница на системата;

2. Избиране на опцията *Vehicles* от страничното меню;
3. Избиране на иконата *Preview* срещу превозното средство от списъка, от чиито пресрочени задачи се интересува потребителят;
4. Маркиране на бутона *Overdue services*;

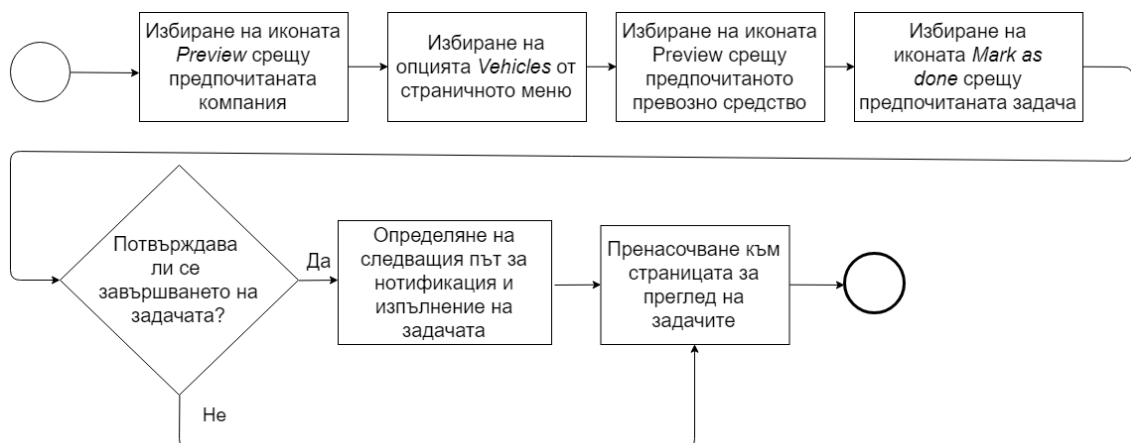


Фигура 28: „Процес по филтрация на просрочените задачи към превозно средство”

4.5.8. Отбелязване на задача като завършена

Извършва се от администратор на компания или гост на компания, вписани в системата. Включва следните стъпки:

1. Избиране на иконата *Preview* срещу компания от списъка с компании от началната страница на системата;
2. Избиране на опцията *Vehicles* от страничното меню;
3. Избиране на иконата *Preview* срещу превозното средство от списъка, към което потребителят иска да маркира задача като завършена;
4. Пренасочване към страницата за преглед на задачи към превозно средство;
5. Избиране на иконата *Mark as done* срещу задачата от списъка, която потребителят иска да маркира като завършена;
6. При натискане на бутона *OK* от прозореца за потвърждение премини към 8, иначе към 7;
7. При натискане на бутона *Cancel* от прозореца за потвърждение премини към 4;
8. Задачата е маркирана като завършена, премини към 4;

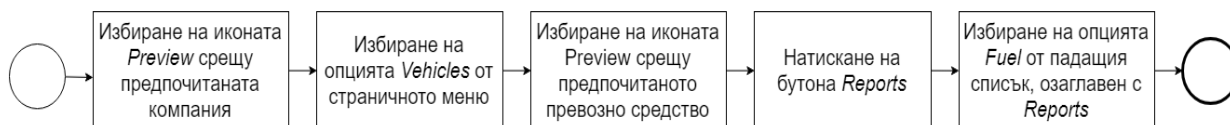


Фигура 29: „Процес по маркиране на задача като завършена”

4.5.9. Преглед на репорт за горивото на превозно средство

Процесът се извършва от администратор на компания или гост на компания, вписани в системата. Той включва следните стъпки:

1. Избиране на иконата *Preview* срещу компания от списъка с компании от началната страница на системата;
2. Избиране на опцията *Vehicles* от страничното меню;
3. Избиране на иконата *Preview* срещу превозното средство от списъка, чийто репорт иска да види;
4. Натискане на бутона *Reports* от менюто с бутони *Services* и *Reports*;
5. Избиране на опцията *Fuel* от падащия списък, озаглавен с *Reports*;
6. Визуализиране на едноседмичен репорт за нивото на горивото на превозното средство;

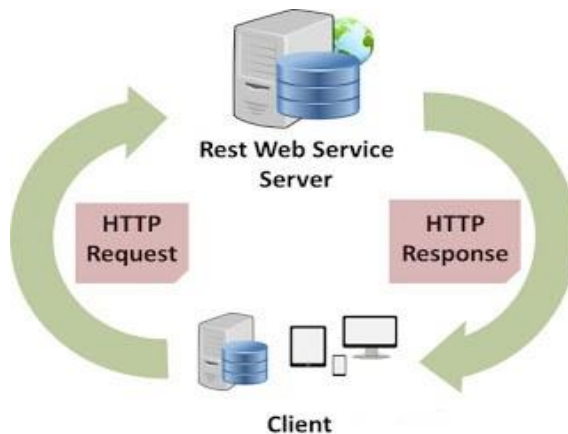


Фигура 30: „Процес по преглед на репорт за горивото на превозно средство”

5. Проектиране

5.1. Обща архитектура [25]

Системата е от тип REST API (*Representational State Transfer API*). Състои се от две основни части – клиент и сървър. Клиентът е фронт-енд частта (*User Interface*), а сървърната част включва бек-енд логиката (API и база от данни). Клиентът и сървърът комуникират помежду си посредством HTTP (*Hypertext Transfer Protocol*) заявки. Най-често използваният език при REST комуникацията е JSON (*Javascript Object Notation*) (Фигура 31).



Фигура 31: „REST комуникация” [24]

Системата се възползва от редица предимства на REST API:

- **Гъвкавост** – Гъвкавостта му позволява вместо JSON лесно да се използват и други нотации за обмен на данни между клиент и сървър като XML (*Extensible Markup Language*) и YAML (*YAML Ain't Markup Language*). Също така евентуална смяна на използвания клиент с друг е напълно възможна.
- **Независимост на заявките** – Заявките към сървъра могат да се извършват независимо една от друга, тъй като всяка от тях съдържа всички данни, необходими за успешно завършване.
- **Надеждност** – REST API разчита единствено на данните, предоставени в резултат на самата комуникация. Идентификационната информация не се съхранява на сървъра при осъществяване на повиквания. Вместо това всяко повикване на сървъра съдържа необходимите данни (например идентификатор за достъп). Това допринася за увеличаване на надеждността на интерфейса. [25]

Клиент-сървър приложенията се разделят на логически части, наречени нива, които изпълняват определена роля. Софтуерната архитектура на системата за управление на флотилия от превозни средства се състои от три нива (*3-tier* архитектура). Те са:

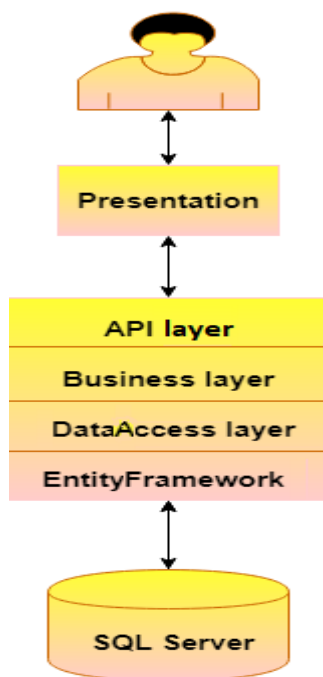
- Презентационно – уеб браузър
- Приложение – логика на системата
- Съхранение – база от данни

Тази архитектура има следните предимства:

- **Сигурност** – Всяко от трите нива може да бъде защитено самостоятелно;
- **Модифицируемост** – Всяко ниво може да се променя без промяната да се отразява на другите;
- **Мащабируемост** – Всяко ниво може да се разширява без това да се отрази на останалите нива; [4]

Сървърната част се състои от пет нива (Фигура 32):

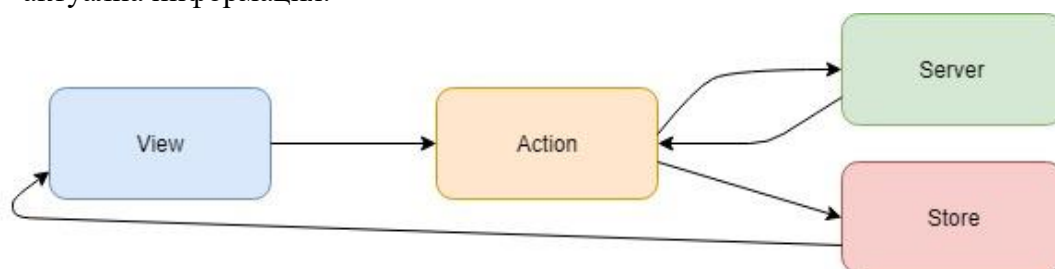
- **API** – Приема заявките на клиента и връща отговора на сървърната част в JSON формат. Извършва се валидация на входните данни, получавани от клиента.
- **Business** – На това ниво е основната логика в сървърната част. Прилагат се бизнес правила върху:
 - съхраняваните данни преди предаването им към API нивото;
 - данните, изпращани от клиента посредством API нивото;
- **DataAccess** – Предоставя опростен достъп до съхраняваните данни в базата чрез класове. Така се създава по-високо ниво на абстракция. Чрез него се извършва връзката с *EntityFramework*.
- **EntityFramework** – Позволява на .NET програмистите да работят с данни от базата чрез обекти без директно да достъпват таблиците в базата.
- **Microsoft SQL Server** – Релационна база от данни;



Фигура 32: „Клиент-сървър архитектура” [25]

При реализацията на клиента е използвана, създадената от *Facebook*, концепция *Flux*. Зад нея стои идеята за еднопосочен поток на данните. Във *Flux* има три основни елемента (Фигура 33):

- **Views** – Това са *ReactJS* компоненти, които се визуализират в браузъра;
- **Actions** – Функции, които се извикват от *View* компонентите при възникването на някакво събитие, например взаимодействие на потребителя с интерфейса. В резултат на това се правят заявки към сървъра.
- **Stores** – В тях се съхраняват върнатите данни от сървъра при изпълнението на успешна заявка. Те се достъпват от *View* компонентите, за да се визуализира актуална информация.



Фигура 33: „Концепцията Flux” [24]

5.2. Файлова структура

Репозиторието на кода в *GitLab* (https://gitlab.com/Monika_Spasova/fleetmanagement) се състои от три папки *Server*, *Client*, и *Resources*. В папката *Resources* се съхраняват документацията към системата и нейният шаблон, използваните изображения и други

помощни материали. Папките *Server* и *Client* съдържат кода съответно на бек-енд и фронт-енд логиката на системата.

Файловата структура на папката *Server* е организирана спрямо основните нива на архитектурата на сървърната част. Тя включва следните директории:

- **Data** – В нея се съдържат моделите, въз основа на които чрез *EntityFramework* се генерира базата от данни;
- **DataService** – Състои се от три поддиректории:
 - **Models** – Тук се намират моделите, съответни на таблиците в базата, които се използват от функциите на *DataService* нивото (*Company.cs*, *Service.cs*, *TelematicsData.cs* и т.н.);
 - **Contracts** – Съдържа интерфейсите на методите от *Service* папката на същото ниво (*IUserDataAccessService.cs*, *ITelematicsDataAccessService.cs* и т.н.);
 - **Service** – Във файловете на тази директория (*UserDataService.cs*, *TelematicsDataAccessService.cs* и т.н.) се намират функциите, извиквани от по-горното бизнес ниво, с цел извършване на различни операции с базата от данни;
- **BusinessService** – Състои се от три поддиректории:
 - **Models** – Тук се намират моделите, които се използват от функциите на *BusinessService* нивото (*Company.cs*, *Service.cs*, *TelematicsData.cs* и други);
 - **Contracts** – Съдържа интерфейсите на методите от *Service* папката (*IUserBusinessService.cs*, *ITelematicsDataBusinessService.cs* и други)
 - **Service** – В нея в различни файлове е разположена бизнес логиката на системата (*UserBusinessService.cs*, *TelematicsDataBusinessService.cs* и други);
- **WebApiService** – Съдържа повече папки и файлове в сравнение с разгледаните досега директории. По-важните от тях са:
 - **Models** – Аналогично тук се намират моделите, използвани от функциите на *WebApiService* нивото;
 - **Controllers** – Съдържа така наречените контролери (*CompanyController.cs*, *TelematicsDataController.cs* и т.н.). Методите на тези файлове се извикват от клиента с цел осъществяване на комуникацията клиент-сървър. Контролерите извикват функциите на по-ниското ниво *BusinessService*.
 - **Web.config** – Съдържа информация за връзката с базата от данни;
- **Infrastructure** – Това е единствената директория, която не представлява ниво от комуникацията между базата от данни и клиента. В папката *JobScheduler* се

намират файлове, които съдържат регулярно изпълнявани задачи като изпращането на имейли и създаването на телематични данни в базата.

- **DataAccessServiceTests** – Съдържа *unit* тестове на *DataAccessService* нивото;
- **BusinessServiceTests** – Съдържа *unit* тестове на *BusinessService* нивото;
- **WebApiServiceTests** – Съдържа *unit* тестове на *WebApiService* нивото;

Основните компоненти на директорията *Client* са:

- **package.json** – съдържа имената на библиотеките, които е необходимо да се инсталират за стартиране на клиента;
- **src** – Тази директория на практика съдържа цялата логика на клиента. Тя съдържа следните поддиректории:
 - **components** – Тук са файловете съответстващи на сегмента *View* от *Flux* концепцията. Тези файлове са с разширение *jsx* и се наричат компоненти. Тяхното съдържание се визуализира на екрана;
 - **actions** – Файловете в тази папка съдържат функции, които се извикват от компонентите. *Action* функциите викат методи в папката *services*;
 - **services** – Извикват методите от контролерите на сървърната част;
 - **stores** – С помощта на *action* функциите отговорът на сървъра се изпраща за съхранение към *stores* методите. Компонентите визуализират данните чрез извикване на *stores* функциите.
 - **styles** – Съдържа *CSS* файловете на системата;
 - **utils** – Във файловете на тази директория се съхраняват някои често използвани функции като тези за валидация и оторизиран достъп;
- **tests** – Съдържа *unit* тестовете на клиентската част;

5.3. База от данни

Името на базата от данни е *FleetManagement*. Нейната колация *Latin1_General_CI_AS* осигурява запазването на данните без да прави разлика между малки и главни букви (*case-insensitive*), но третира символи с и без ударение като различни (*accent-sensitive*). Таблиците *AspNetRoles*, *AspNetUserClaims*, *AspNetUserLogins*, *AspNetUserRoles*, *AspNetUsers* са автоматично генерирани при конфигурациите на библиотеката *ASP.NET Identity* в *Microsoft Web Api 2* проекта (Фигура 34).

В таблицата *AspNetUsers* се съхраняват имейла, потребителското име и хешираната парола на регистрирания потребител. Останалите колони в нея засега не се използват в разработваната система. Таблиците *AspNetUsers* и *Companies* са свързани чрез таблицата *UserCompanies*, която има само три атрибута *Id*, *UserId* и *CompanyId*. Чрез тях се осъществява връзката много към много – една компания може да има много потребители и един потребител може да има много компании. Таблицата *UserCompanies* се използва от функционалността за осигуряване на достъп на потребители до определена компания.

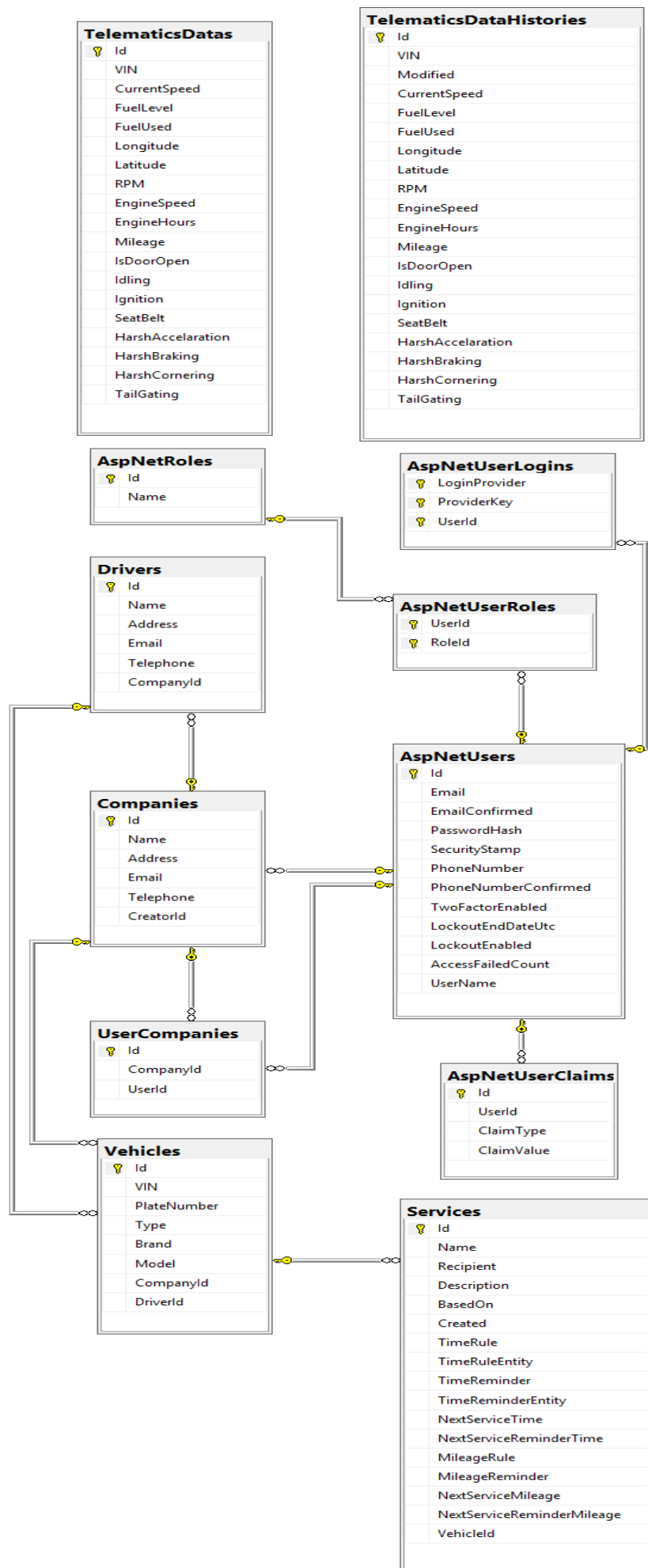
Между таблиците *Companies* и *AspNetUsers* се осъществява още една връзка. Това става чрез атрибута *CreatorId* на таблицата *Companies*, който е външен ключ към таблицата *AspNetUsers*. Тази връзка се използва за определяне на ролята на потребителя според компания – администратор или гост.

Таблиците *Drivers* и *Vehicles* са свързани с *Companies* чрез външен ключ *CompanyId*. Благодарение на тези връзки е възможно една компания да има много шофьори и много превозни средства.

Съществува връзка между *Drivers* и *Vehicles* чрез външния ключ *DriverId* от *Vehicles*. Това позволява свързването на превозно средство със създаден в системата шофьор. Напълно очаквано и таблицата *Services* има външен ключ към *Vehicles*. Така се осъществява създаването на задачи към превозно средство.

В таблицата *TelematicsDatas* се съхранява информацията за текущите характеристики на превозното средство. *TelematicsDataHisories* се използва за съхранение на данните, които се визуализират в едноседмичния репорт към превозно средство. Таблиците *TelematicsDatas* и *TelematicsDataHisories* се свързват с таблицата *Vehicles* по атрибута VIN. Таблицата *TelematicsDataHisories* се попълва всеки път при въвеждане на данни в *TelematicsDatas*. Единствената разлика в схемата на двете таблици е колоната *Modified* в *TelematicsDataHisories*, в която се съхранява времето, към което телематичните данни са актуални.

Таблиците *TelematicsDatas* и *TelematicsDataHisories* имат много атрибути, които не се използват в текущата версия на системата. В бъдеще съхраняваната в тях информация може да се визуализира в модула с репорти, както километража и нивото на горивото в сегашната версия. Такива са колоните за текуща скорост, географска ширина и дължина, скорост на двигателя, часове на двигателя, обороти, а също и тези, които индикират за запалване на автомобила, престой, отворена врата, поставен колан, рязко спиране или ускорение, неспазване на дистанция.

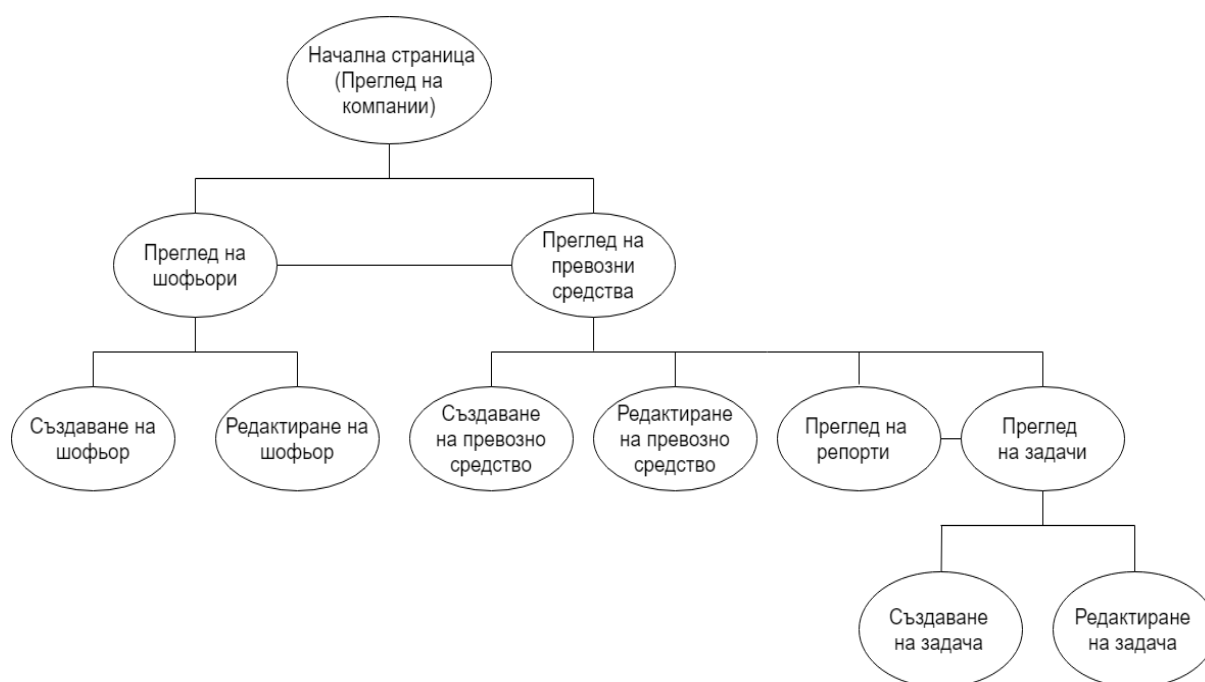


Фигура 34: „Диаграма на базата от данни”

5.4. Информационна архитектура [37]

Информационната архитектура на системата е от вида *Co-existing hierarchy*, която разширява строго йерархичната архитектура. Характерно за нея е наличието на начална страница на върха на йерархията. При тази информационна архитектура достъпът до дадена страница е възможен чрез (Фигура 34):

- родителската страница;
- подстраница на същата родителска страница (което е невъзможно при строго йерархичната архитектура);



Фигура 35: „Информационна архитектура на системата”

5.5. Диаграми

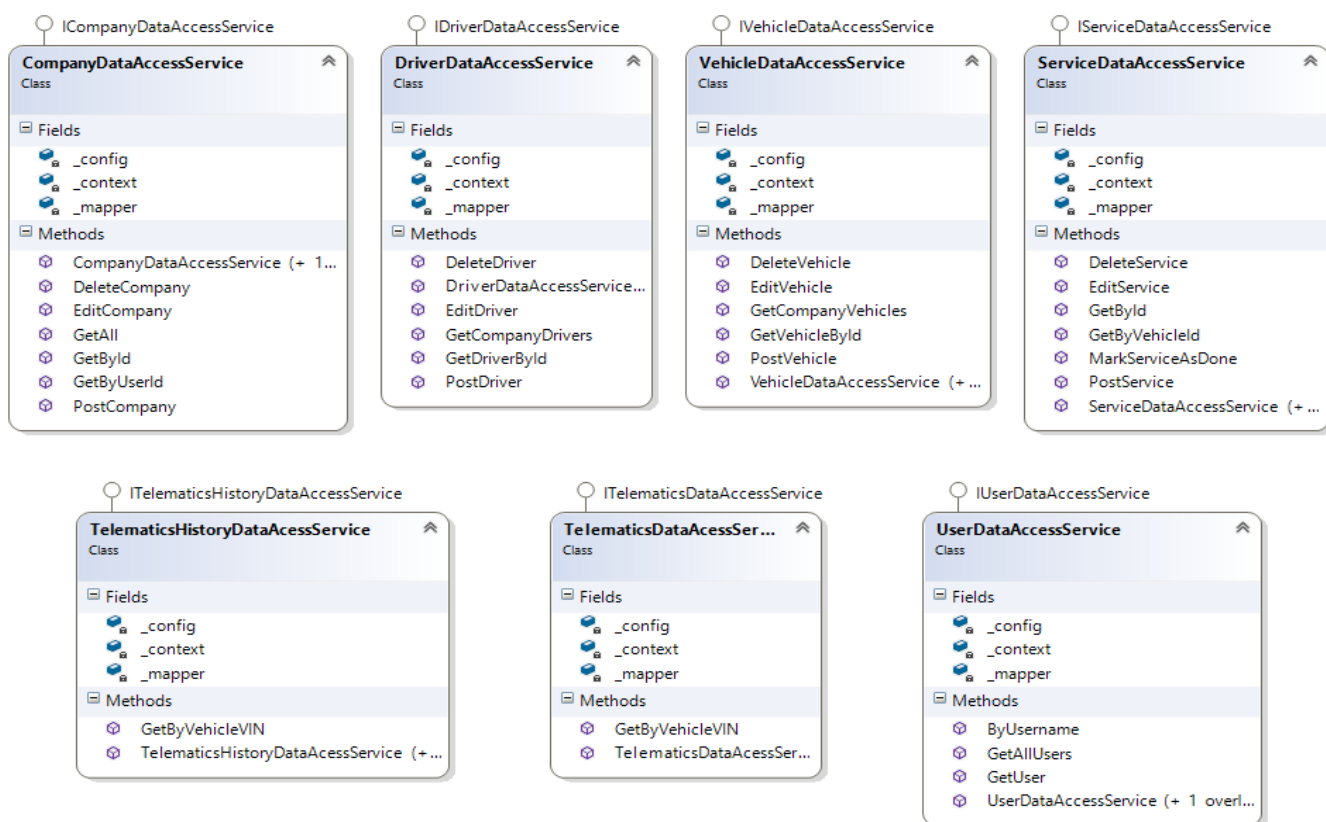
На Фигура 36 е представена диаграма на класовете на ниво *DataAccessService*. Показани са класовете и интерфейсите, които те имплементират. Полетата на всички класове на това ниво са:

- *_config* и *_mapper* – използват се за преобразуването на класовете между *DataAccessService* и по-ниското *Data* ниво. Това позволява превръщането на *EntityFramework* класовете в *DataAccessService*, за да могат те да бъдат връщани като такива от *DataAccessService* методите.
- *_context* – това е инстанция на класа *FleetManagementDbContext*. Използва се за извършването на операции с базата от данни.

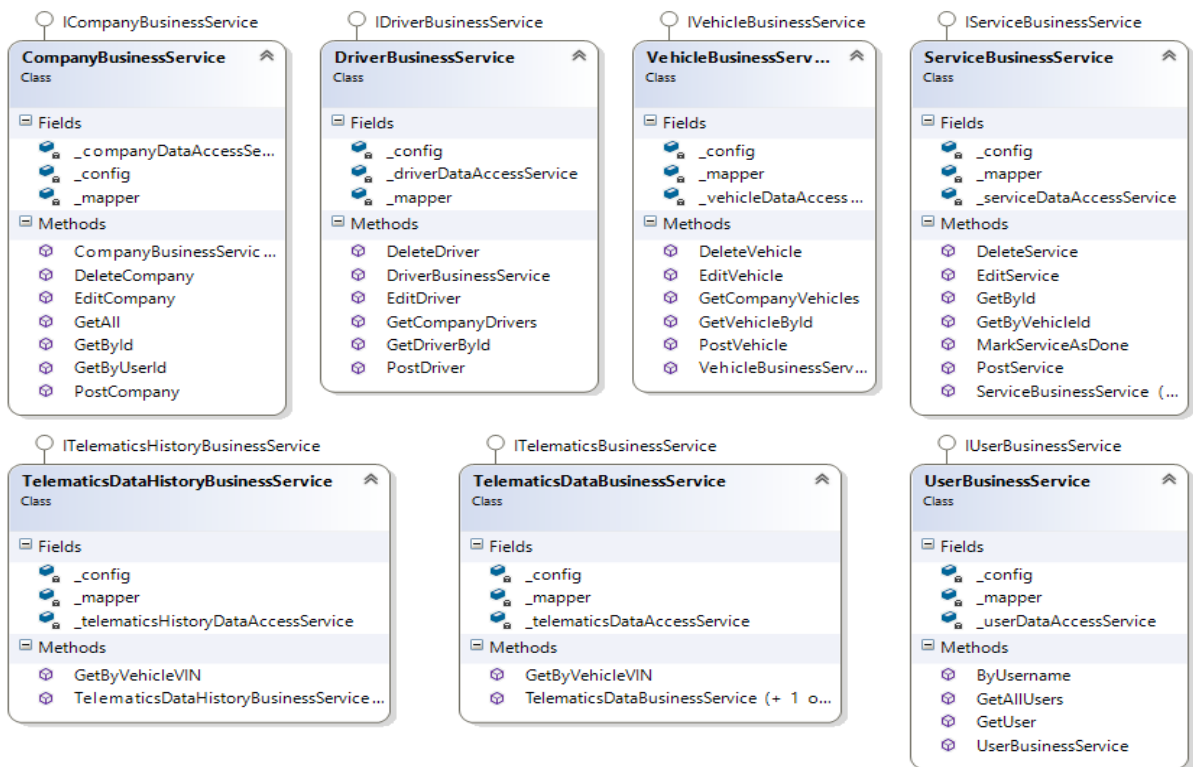
На Фигура 37 е изобразена диаграма на класовете на *BusinessService* нивото. Вижда се, че методите на класовете са идентични с тези в *DataAccessService*. Полетата *_config* и *_mapper* служат за преобразуването на *DataAccessService* класовете в *BusinessService*

класове. Единствената разлика е в имената на наследените интерфейси и, че при *BusinessService* нивото вместо *_context* полето има инстанции на съответните класове от по-ниското *DataAccessService* ниво (*_companyDataAccessService*, *_driverDataAccessService* и т.н).

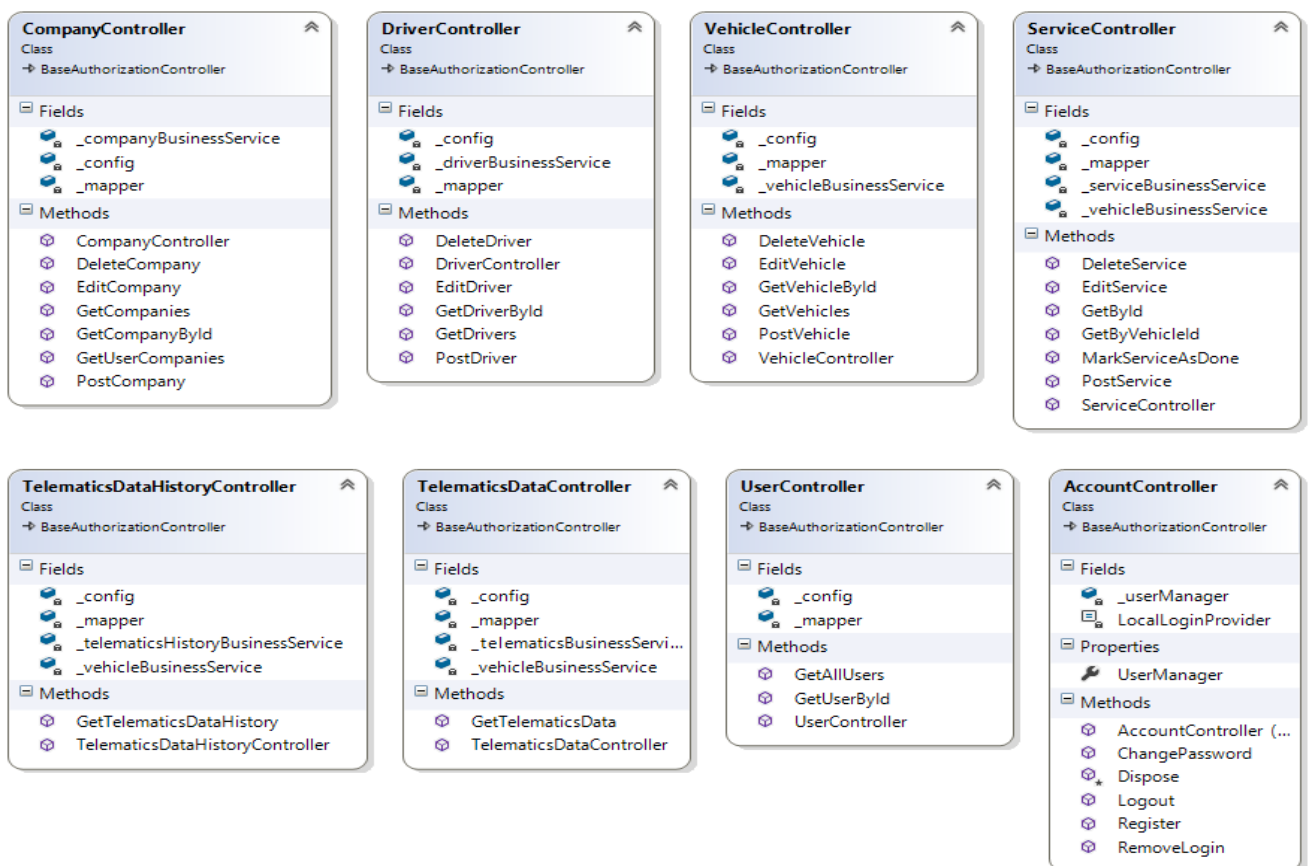
Диаграмата на класовете на *WebApiService* нивото е основана на същия принцип (Фигура 38). Полетата *_config* и *_mapper* служат за преобразуването на *BusinessService* класовете в *WebApiService* класове. Всеки клас съдържа и инстанция на съответния *BusinessService* клас (*_companyBusinessService*, *_telematicsBusinessService* и т.н.). Специфично за *WebApiService* нивото е, че класовете (контролерите) не наследяват интерфейси, както при *DataAccessService* и *BusinessService*.



Фигура 36: „Класова диаграма на *DataAccessService* нивото”

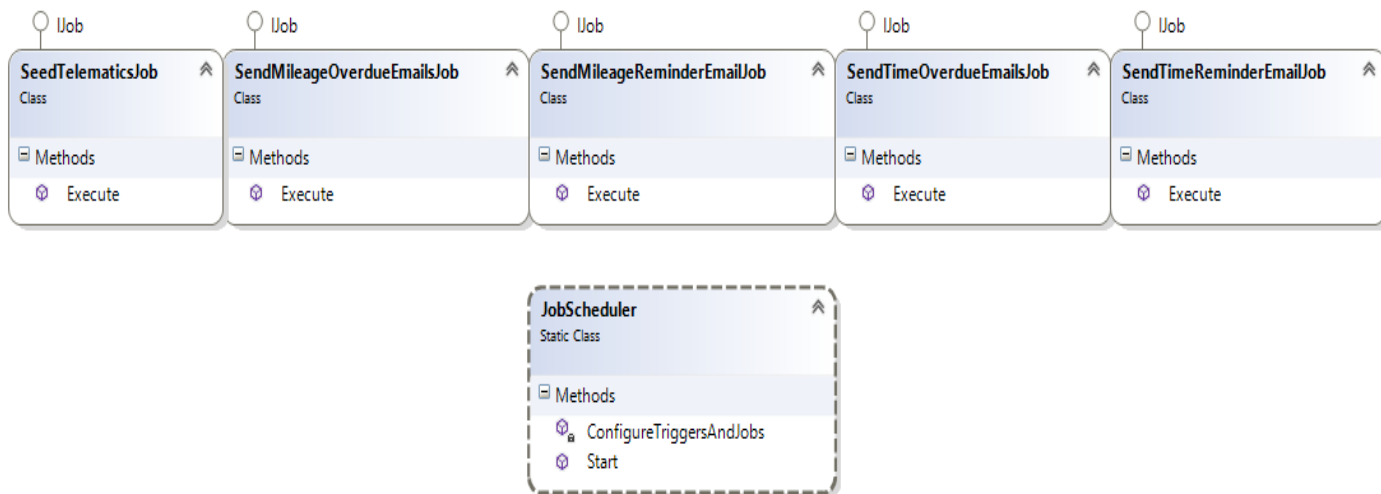


Фигура 37: „Класова диаграма на BusinessService нивото”



Фигура 38: „Класова диаграма на WebApiService нивото”

На Фигура 39 са изобразени класовите диаграми на регулярно изпълняваните задачи чрез библиотеката *Quartz*.



Фигура 39: „Класова диаграма на класовете в директорията *Infrastructure/JobScheduler/Jobs*”

Всички класове съдържат методи *Execute*, които с помощта на класовете в директорията *Infrastructure/Helpers*, извършват изпращането на имейли и генерирането на телематични данни. В метода *ConfigureTriggersAndJobs* на класа *JobScheduler* се определя честотата на изпълнение на задачите. Така изглежда кодът за извикване на задачата *SendTimeOverdueEmailsJob*:

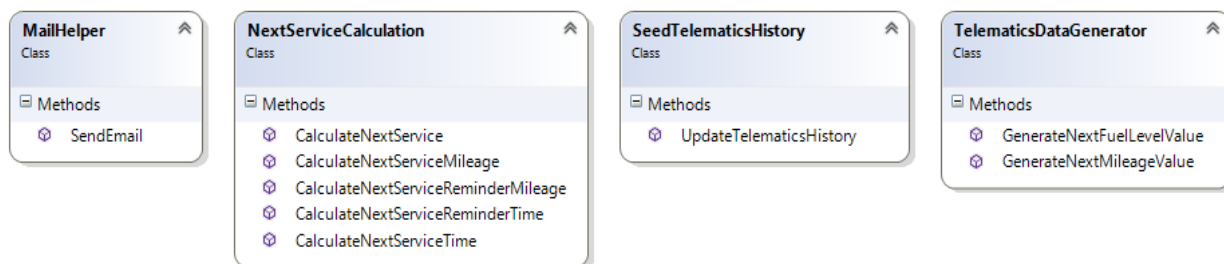
```

.....
var sendTimeOverdueEmailsJob = JobBuilder.Create<SendTimeOverdueEmailsJob>().Build();
var sendTimeOverdueEmailsJobTriggers = new List<ITrigger>
{
    TriggerBuilder.Create().WithDailyTimeIntervalSchedule(
        s => s.WithIntervalInHours(24).OnEveryDay()
            .StartingDailyAt(TimeOfDay.HourAndMinuteOfDay(0,0))).Build()
};

triggersAndJobs.Add(sendTimeOverdueEmailsJob, sendTimeOverdueEmailsJobTriggers);
.....

```

На Фигура 40 са изобразени класовите диаграми на помощните класове, извиквани от задачите в директорията *Infrastructure/JobScheduler/Jobs*.



Фигура 40: „Класова диаграма на класовете в директорията *Infrastructure/Helpers*”

6. Реализация и тестване

6.1. Реализация на модулите

6.1.1. Вписване и регистрация

От страна на клиента при попълване на регистрационната форма и формата за вписване се използват методите на класа *AuthorizationService* в директорията *Client/src/services/AuthorizationService.js*. Те правят заявки към сървъра:

```
class AuthorizationService
{
    static registerUser(newUser)
    {
        return axios.post('http://localhost:19631/api/account/register', newUser);
    }

    static loginUser(user)
    {
        axios.defaults.headers = {
            'Content-Type': 'application/x-www-form-urlencoded',
            'Accept': 'application/json',
        };

        var parsedData = Object.keys(user).map(function(key) {
            return encodeURIComponent(key) + '=' + encodeURIComponent(user[key])
        }).join('&');

        return axios.post('http://localhost:19631/token', parsedData);
    }
}
```

В сървърната част при регистрация тези заявки извикват метода *Register* на класа *AccountController* (*Server/WebApiService/Controllers/AccountController.cs*). При вписване в сървъра се използва класа *OAuthAuthorizationServerOptions* от пакета *Microsoft.Owin.Security*. Инстанция на този клас се създава в *Startup* класа в *Server/WebApiService/App_Start/Startup.Auth.cs*. Там се определя пътът *TokenEndpointPath* от *Microsoft.Owin.Security*, към който клиентът да изпраща заявки при вписване на потребителя чрез метода:

```
public void ConfigureAuth(IAppBuilder app)
{
    .....

    OAuthOptions = new OAuthAuthorizationServerOptions
    {
        TokenEndpointPath = new PathString("/Token"),
        Provider = new ApplicationOAuthProvider(PublicClientId),
        AccessTokenExpireTimeSpan = TimeSpan.FromDays(2),
        AllowInsecureHttp = true
    };

    .....
}
```

При успешна заявка от клиента (валидно потребителско име и парола) този метод връща токен. Него клиентът съхранява в брауъра:

```
const setLocalStorageItems = response => {
    localStorage.setItem('token', response.data.access_token);
    localStorage.setItem('expiration', response.data['.expires']);
    localStorage.setItem('userId', response.data.user_id);
}
```

Впоследствие го използва за осигуряване на оторизиран достъп до функционалностите на приложението:

```
static async getDriver(driverId)
{
    if (isLoggedIn())
    {
        return await axios.get(`http://localhost:19631/api/drivers/${driverId}`);
    }

    logout();
}
```

Функцията *isLoggedIn* проверява дали в брауъра се съхранява токен и дали той не е изтекъл.

6.1.2. Генериране на телематични данни

За целите на дипломната работа е създадена функционалност за генериране на телематични данни. Това става чрез задача *SeedTelematicsJob*, изпълнявана на всеки 60 минути. При нейното изпълнение за всяко превозно средство в базата се обновяват стойностите *Mileage* и *FuelLevel* в таблицата *TelematicsData*, а също така обновените стойности се записват в таблицата *TelematicsDataHistory*. Ако превозното средство е новосъздадено и все още няма телематични данни, се създава нов запис в таблицата *TelematicsData*. Кодът за тази функционалност се намира в директорията *Server/Infrastructure/JobsScheduler/Jobs/SeedTelematicsJob.cs*:

```
.....
var vehicles = await dbContext.Vehicles.ToListAsync();
foreach (var vehicle in vehicles)
{
    var telematicsData = await dbContext.TelematicsDatas
        .FirstOrDefaultAsync(t => t.VIN == vehicle.VIN);
    TelematicsData newTelematicsData = new TelematicsData
    {
        VIN = vehicle.VIN,
        Mileage = TelematicsDataGenerator.GenerateNextMileageValue(telematicsData?.Mileage),
        FuelLevel = TelematicsDataGenerator.GenerateNextFuelLevelValue(telematicsData?.FuelLevel)
    };

    if (telematicsData == null)
    {
        dbContext.TelematicsDatas.Add(newTelematicsData);
    }
    else
    {
        telematicsData.Mileage = newTelematicsData.Mileage;
        telematicsData.FuelLevel = newTelematicsData.FuelLevel;
    }
}
```

```
await SeedTelematicsHistory.UpdateTelematicsHistory(newTelematicsData, dbContext);
}

await dbContext.SaveChangesAsync();
```

Създадени са и алгоритми, чрез които да се генерират валидни стойности за километража (постоянно нарастващ) и нивото на горивото. Нивото на горивото се понижава до ниски стойности, след което шофьорът зарежда и то се покачва отново. Алгоритмите са в класа *TelematicsDataGenerator* в *Server/Infrastructure/Helpers/SeedTelematicsGenerator.cs*:

```
public class TelematicsDataGenerator
{
    public static int? GenerateNextMileageValue(int? currentMileage)
    {
        Random random = new Random();
        int maxDifference = 10;

        var startValue = currentMileage ?? random.Next(100, 200);
        var nextValue = random.Next(startValue, startValue + maxDifference);

        return nextValue;
    }

    public static int? GenerateNextFuelLevelValue(int? currentFuelLevel)
    {
        Random random = new Random();
        var difference = 5;

        var criticalFuelLevel = random.Next(10, 30);
        var normalFuelLevel = random.Next(50, 80);

        var startValue = currentFuelLevel ?? random.Next(criticalFuelLevel, 100);
        var nextValue = startValue <= criticalFuelLevel
            ? random.Next(normalFuelLevel, 100)
            : random.Next(startValue - difference, startValue);

        return nextValue;
    }
}
```

6.1.3. Използване на телематично оборудване

В реалността приложението ще бъде използвано с данни от интегрирано в автомобилите телематично оборудване. За целта системата ще прави заявки към АПИ-тата на доставчици на телематични устройства на всеки 20 секунди, използвайки VIN номерата на превозните средства. Като резултат АПИ-тата на производителите ще връщат телематичните данни за всеки автомобил в XML формат към системата. Системата ще очаква данните в стандартизирания АЕМ/АЕМР формат, което ще позволи лесна интеграция с голям брой производители на телематично оборудване. Единствената разлика между различните производители, която засяга разработваната система, се изразява в различните URL-и на техните АПИ-та. Благодарение на АЕМ/АЕМР формата на данните, тяхната по-нататъшна обработка ще бъде еднаква. Това ще се извършва чрез задача подобна на *SeedTelematicsJob*, която в текущата

версия на системата генерира тестови телематични данни. Всъщност основната разлика с нея ще бъде начинът, по който системата получава данните за километража и нивото на горивото – в настоящата версия чрез методите на класа *TelematicsDataGenerator*, а в бъдещата чрез заявки към АПИ-тата на производителите на телематично оборудване. За постигане на добра ефективност е възможно добавянето на колона *TelematicsProvider* към таблицата *Vehicles*, в която да се съхранява името на доставчика на телематично оборудване. Представям псевдокод на задача, която прави регулярни заявки към АПИ-тата на доставчиците на телематично оборудване:

```
.....
var vehicles = await dbContext.Vehicles.ToListAsync();
foreach (var vehicle in vehicles)
{
    var response;
    if (vehicle.TelematicsProvider == "StyleTronic")
    {
        response = Do a request to the StyleTronic's API
    }
    else if (vehicle.TelematicsProvider == "GeoTab")
    {
        response = Do a request to the GeoTab's API
    }
    else if (.....)
    {
        .....
    }

    var mileage = response["Mileage"];
    var fuelLevel = response["FuelLevel"];

    var telematicsData = await dbContext.TelematicsDatas
        .FirstOrDefaultAsync(t => t.VIN == vehicle.VIN);
    TelematicsData newTelematicsData = new TelematicsData
    {
        VIN = vehicle.VIN,
        Mileage = mileage,
        FuelLevel = fuelLevel
    };

    if (telematicsData == null)
    {
        dbContext.TelematicsDatas.Add(newTelematicsData);
    }
    else
    {
        telematicsData.Mileage = mileage;
        telematicsData.FuelLevel = fuelLevel;
    }

    await SeedTelematicsHistory.UpdateTelematicsHistory(newTelematicsData, dbContext);
}

await dbContext.SaveChangesAsync();
```

6.1.4. Изпращане на имейли за предстоящи и просрочени задачи

Друга важна функционалност, на чиито код си струва да обърнем внимание, е тази за изпращане на имейли за предстоящи и просрочени задачи. Същинското изпращане на имейлите се извършва от създадения за системата *gmail* акаунт

fleet.management.official@gmail.com чрез метода *SendEmail* на класа *MailHelper* в *Server/Infrastructure/Helpers/MailHelper.cs*:

```
public class MailHelper
{
    public static async Task SendEmail(string recipients, string subject, string body)
    {
        var userName = "fleet.management.official@gmail.com";
        var password = ".....";

        var credentials = new NetworkCredential(userName, password);
        var client = new SmtpClient("smtp.gmail.com", 587)
        {
            Credentials = credentials,
            EnableSsl = true
        };

        var mailMessage = new MailMessage(userName, recipients, subject, body);
        await client.SendMailAsync(mailMessage);
    }
}
```

Този метод се извиква от четирите задачи за изпращане на имейли в директорията *Server/Infrastructure/JobScheduler/Jobs*:

- *SendMileageReminderEmailJob.cs*
- *SendTimeReminderEmailJob.cs*
- *SendMileageOverdueEmailJob.cs*
- *SendTimeOverdueEmailJob.cs*

Задачите *SendTimeReminderEmailJob* и *SendTimeOverdueEmailJob* сравняват настоящата дата с датите, записани в колоните съответно *NextServiceReminderTime* и *NextServiceTime*, на базирани на време дейности по обслужване на превозни средства (*BasedOn* == 0). При *SendTimeOverdueEmailJob* имейли се изпращат за дейности, при които настоящата дата е по-голяма от *NextServiceTime*. Представям кода на *SendTimeReminderEmailJob*:

```
public class SendTimeReminderEmailJob : IJob
{
    public async Task Execute(IJobExecutionContext context)
    {
        using (FleetManagementDbContext dbContext = new FleetManagementDbContext())
        {
            var currentTime = await dbContext.Database
                .SqlQuery<DateTime>("SELECT GETUTCDATE()").FirstOrDefaultAsync();

            var todaysDate = new DateTimeOffset(new DateTime(currentTime.Year,
                currentTime.Month, currentTime.Day, currentTime.Hour, currentTime.Minute,
                currentTime.Second, DateTimeKind.Utc));

            var services = dbContext.Services.Where(s =>
                (s.BasedOn == 0 && s.NextServiceReminderTime != null &&
                DbFunctions.TruncateTime(s.NextServiceReminderTime) ==
                DbFunctions.TruncateTime(todaysDate))).ToList();

            foreach (var service in services.ToList())
            {

```

```

        MailHelper.SendEmail(service.Recipient, "Reminder services", $"The
        service {service.Name} for vehicle {service.Vehicle.Brand} with plate
        number {service.Vehicle.PlateNumber} is
        following.").RunSynchronously();
    }
}
}

```

Задачите *SendMileageReminderEmailJob* и *SendMileageOverdueEmailJob* сравняват текущия километраж със стойностите, записани в колоните съответно *NextServiceReminderMileage* и *NextServiceMileage*, на базирани на километраж дейности по обслужване на превозни средства (*BasedOn == 1*). При *SendMileageReminderEmailJob* имейли се изпращат за задачи, при които $NextServiceReminderMileage \in [vehicleTelem.Mileage - 50, vehicleTelem.Mileage + 50]$. При *SendMileageOverdueEmailJob* имейли се изпращат за дейности, при които настоящия километраж е по-голям от *NextServiceMileage*. Представям кода на *SendMileageOverdueEmailJob*:

```

public class SendMileageOverdueEmailsJob : IJob
{
    public async Task Execute(IJobExecutionContext context)
    {
        using (FleetManagementDbContext dbContext = new FleetManagementDbContext())
        {
            var vehiclesVIN = dbContext.Vehicles.Select(v => v.VIN).ToList();

            foreach (var vehicleVIN in vehiclesVIN)
            {
                var vehicleTelematics = await dbContext.TelematicsDatas
                    .FirstOrDefaultAsync(t => t.VIN == vehicleVIN);
                var vehicle = await dbContext.Vehicles
                    .FirstOrDefaultAsync(v => v.VIN == vehicleVIN);

                var services = dbContext.Services.Where(s => s.BasedOn == 1 &&
                    s.NextServiceMileage != null && s.NextServiceMileage <
                    vehicleTelematics.Mileage).ToList();

                foreach (var service in services)
                {
                    MailHelper.SendEmail(service.Recipient, "Overdue service", $"The
                    service {service.Name} for vehicle {vehicle.Brand} with plate number
                    {vehicle.PlateNumber} is overdue.").RunSynchronously();
                }
            }
        }
    }
}

```

6.1.5. Изчисляване на времето/километража за следваща задача и напомняне за задача

Логиката за изчисляване на времето/километража за извършване на задачи и напомняне за предстоящи задачи се изчислява при създаване на задача, редактиране на задача или при маркиране на задача като завършена. И на трите места това е реализирано чрез методите на класа *NextServiceCalculation* в директорията *Server/Infrastructure/Helpers/NextServiceCalculation.cs*. Методът *CalculateNextService*

изчислява времето/километража за извършване на задачите и напомняне чрез помощните функции в същия клас:

- *CalculateNextServiceTime*
- *CalculateNextServiceReminderTime*
- *CalculateNextServiceMileage*
- *CalculateNextServiceReminderMileage*

6.1.6. Модули за компании, шофьори, превозни средства и задачи

Модулите за компании, шофьори, превозни средства и задачи в сървърната част са реализирани изцяло чрез използването на четирите нива от архитектурата – *WebApiService*, *BusinessService*, *DataAccessService* и *Data*. Всяко от първите три нива разполага с *Get*, *Post*, *Put* и *Delete* методи, участващи в извършването на съответните операции. В *DataAccessService* нивото чрез *Data* (*EntityFramework*) моделите се изпълняват операциите в базата от данни.

6.2. Unit тестване

За тестването на клиентската и сървърната част са конфигурирани библиотеки за *unit* тестване (тестване на най-малките единици от кода). Този вид тестване се извършва преди другите видове тестове. То има следните предимства:

- Ако тестовете са валидни и се пускат всеки път при промяна в кода, потенциални проблеми ще бъдат открити.
- Разходите за поправяне на проблеми, забелязани с *unit* тестове, са по-малки в сравнение с откриването и поправянето им на по-късен етап от живота на софтуера.
- Лесни са за дебъгване.

6.2.1. Unit тестове на клиентската част

За създаването на клиентските *unit* тестове са използвани следните технологии:

- *Karma* – Това е така наричаният *test runner*. Той осигурява необходимата среда за изпълнение на тестовете. Неговите конфигурации са във файла *Client/tests/karma.conf.js*. Там се определя на кой порт и браузър се изпълняват тестовете, кои са използваните библиотеки за тестване и много други настройки.
- *Mocha* – Фреймворк за създаване на *JavaScript unit* тестове. Той предоставя необходимия синтаксис за писане на тестовете.
- *Enzyme* – Библиотека за тестване на *ReactJS* компоненти;
- *Sinon* – Използва се за симулиране на поведението на различни функции в контекста на тестовете.

Клиентските тестове се намират в директорията *Client/tests* в различни директории според вида на тествания файл – *components*, *actions* и *stores*. Могат да бъдат стартирани с командата *npm test* в директорията *Client*.

6.2.2. Unit тестове на сървърната част

За създаването на сървърните *unit* тестове са използвани следните технологии:

- *Xunit.net* – Инструмент за писане на *unit* тестове на *.NET Framework*. Той включва:
 - *Test runner* – Чрез него се изпълняват тестовете и се правят репорти с резултатите от тях;
 - *Assertions* – Чрез тях с различни методи се сравняват резултатите от извикването на дадена функция с тестови данни и очакваните резултати.
- *Moq* – Тази библиотека се използва за симулиране на резултатите от функции, използвани в тестовите методи.

Направени са конфигурации за *unit* тестване на *DataAccessService*, *BusinessService* и *WebApiService* нивата. За тестовете на различните нива са създадени съответно проектите – *WebApiServiceTests*, *BusinessServiceTests*, *DataAccessServiceTests*.

6.3. Планиране на тестването

Проверени са основни тестови сценарии за компоненти и функционалности, засягащи всички модули от приложението. Такива са потребителският интерфейс, сигурността, валидацията на въведените данни и други. С тестовите сценарии се означава какво в системата да бъде тествано. Как ще се извърши самото тестване на основните модули е описано по-подробно чрез така наречените *test cases*.

6.3.1. Тестови сценарии

Тестовите сценарии са оформени в табличен вид и са представени в Приложение 1. Освен самия сценарий всеки ред може да съдържа още:

- **Резултат** – Това е резултатът от изпълнението на тестовия сценарий. Той е успешен, ако при проверката не са открити несъответствия. В противен случай е неуспешен.
- **Статус** – Използва се за описание на състоянието на проблема. Приема стойности *разрешен* и *неразрешен*. За успешно изпълнените тестови сценарии статусът е *разрешен*. Ако изпълнението на сценарият е довел до откриването на проблеми (завършил е с неуспех), то статусът може да има две стойности:
 - **Разрешен** – Ако след откриването на проблема, той е отстранен;
 - **Неразрешен** – Ако след откриването на проблема, върху отстраняването му все още не е работено;

На Таблица 1 са показани тестовите сценарии, засягащи формите на системата, техните валидации и прозорци за потвърждение. Сценариите в Таблица 2 са насочени към потребителския интерфейс, а тези в Таблица 4 към сигурността.

Изпратените имейли за предстоящи и пресрочени задачи, базирани на време или километраж, също трябва да имат унифициран вид. На Таблица 3 са показани основните тестови сценарии, проверени за четирите вида имейли.

6.3.2. Стъпки за тестване (*test cases*)

Стъпките за тестването на основните функционалности на приложението ще бъдат описани по-подробно чрез *test cases*. Докато сценариите указват какво ще бъде тествано, чрез *test cases* се задават конкретни стъпки за тестване. Един тестови сценарий може да включва няколко *test cases* с различни входни данни. Създадените *test cases* трябва да бъдат изпълнявани последователно, защото за изпълнението на всеки от тях са нужни данни, генерирани в предходните. В Приложение 2 в табличен вид са показани основните за системата *test cases*.

6.3.3. Анализ на резултатите от тестването

Извършеното тестване показва, че приложението е стабилно. При изпълнението на различните *test cases* не са открити сериозни проблеми, свързани с работата и сигурността на системата. Локализиран са бъгове при:

1. Валидацията на някои полета в различни форми на системата:
 - Валидацията на полетата за правилен формат на имейл и телефонен номер във формите на системата е пропусната.
 - Не се появява съобщение за грешка при неправилно потвърждение на паролата или при опит за регистрация със съществуващо потребителско име или имейл;
 - Не се визуализира съобщение за грешка при опит за вписване с неправилен имейл или парола;
2. Многократното натискане на бутона *Save* на формите – Това води до създаване на голям брой идентични записи в системата. Очаква се бутонът *Save* да стане неактивен след първото му натискане, за да се предотврати възможност за неколнократно натискане.
3. Валидацията за положителни стойности на полетата *Time Rule*, *Time Reminder*, *Mileage Rule* и *Mileage Reminder* във формата за създаване и редактиране на задача – Не се появява съобщение за грешка при въвеждане на отрицателна стойност.
4. Опит за взаимодействие със системата с изтекъл оторизационен токен – Потребителят не се пренасочва към формата за вписване, както се очаква. Вместо това на екрана се визуализира празна страница.

Споменатите проблеми са поправени след откриването им. Затова и техният статус в таблиците с тестовите сценарии е „Разрешен”.

6.4. Конфигурации за локално стартиране на системата

Приложението е разработено на операционна система *Microsoft Windows 10*. За локалното му стартиране е необходима инсталацията на следните програми:

1. Сървър:

- *Microsoft SQL Server Management Studio* – За разработката е използвана версия 15;
- *Microsoft Visual Studio* – За разработката е използвано *Microsoft Visual Studio 2015 Enterprise*;

2. Клиент:

- *Node.js* – За разработката е използвана версия 6.10.0;
- Едитор – Той е необходим за отваряне на кода на клиента. Липсата му няма да попречи на работата на системата. При разработката е използван *Microsoft Visual Studio Code 1.4.0*;

Най-напред е необходимо да се създаде локална база. За целта трябва да се извършат следните стъпки:

- В *Microsoft Visual Studio* се отваря *fleetmanagement/Server/FleetManagement.sln*;
- Във файла *fleetmanagement/Server/WebApiService/Web.config* в тага *connectionstring* в атрибута *connectionstring* е необходимо да се обнови стойността на *Data Source*. Като стойност на *Data Source* трябва да се постави името на инсталирания SQL сървър. Направените промени се запазват.
- Отваря се *Package Manager Console* и от падащия списък озаглавен с *Default project* се избира стойността *WebApiService*. Тогава в конзолата се изпълнява командата *Update-Database*. След успешното изпълнение на командата в *SQL Server Management Studio* се създава нова база *FleetManagement*.

След това трябва да се стартира *WEB API* проекта. За целта:

- От менюто *Solution Explorer* с десен бутон се кликва върху проекта *WebApiService* и се избира опцията *Set as StartUp Project*.
- От главното меню на *Visual Studio* чрез падащ списък може да се промени браузъра, на който да се изпълни сървърната част. Проектът се стартира с кликане върху зелената стрелка до него. При успешно стартиране в избрания браузър се отваря прозорец на *http://localhost:19631/*.

На последно място е необходимо да се стартира и клиентската част. Преди това трябва да се инсталират необходимите пакети. Това става чрез изпълнението на командата *npm install* в *CMD* конзолата в директорията *fleetmanagement/Client*, където се намира файлът *package.json*. При успешното изпълнение на командата в *fleetmanagement/Client* се създава папка *node_modules*, съдържаща инсталираните пакети. След това клиентът се стартира чрез командата *npm start* в същата директория. При успех се отваря прозорец в *Google Chrome* на *http://localhost:3000/* и се визуализира формата за вписване в системата.

6.5. Хостване

Приложението ще бъде деплойнато в платформата <https://appharbor.com>. След създаване на акаунт в системата тя предоставя безплатни хостинг услуги. За успешно конфигуриране трябва да се използва предоставеният от интерфейса на *AppHarbor* адрес за достъп на техния *SQL Server (Connection String)*. След това се въвежда името на домейна, което ще бъде свързано с предложението от *AppHarbor* свободен IP адрес. След избиране на опцията *Repository URL* платформата осигурява линк за качване на системата. За успешно деплойване е необходимо чрез *Git* команда да се качи сорс кода на приложението на линка на репозитория (*Repository URL*). След успешно качване и билд приложението се стартира. То може да бъде достъпно от въведения при конфигурацията домейн.

7. Заключение

7.1. Обобщение на изпълнението на началните цели и претенциите за оригинални резултати

Управлението на флотилия от превозни средства е сложен процес. Той се основава предимно на телематични данни. Успешният *Fleet Management* софтуер използва тези данни, за да проследява, анализира и работи за оптимизирането на различните аспекти от процеса. Използването на телематика в компании с малки автопаркове все още не е широко застъпено. Според статия на *Financial Times*, обаче автомобилните производители ще се възползват от икономията от мащаба, за да предлагат автомобили с вградено телематично оборудване. Въз основа на това те прогнозираят, че в близките от пет до седем години телематичното оборудване ще стане неразделна част от всички превозни средства. [38] В подобна обстановка на пазара ще има голям брой производители на автомобили с вградено телематично оборудване и в същото време голям брой странични производители на такова оборудване. Тогава нуждата от лесна интеграция между различните видове софтуер и хардуер значително ще нарастне. За да се осигури съвместимост, ще са необходими системи, работещи съгласно общоприетия стандарт за единен формат на телематичните данни АЕМ/АЕМР. Именно първообраз на такава система беше създаден в рамките на настоящата дипломна работа.

Изложението на дипломната работа и създаването на системата са базирани на основните етапи за разработка на софтуер. Проведеното тестване показва, че приложението е стабилно и че работи съгласно дефинираните работни процеси. По време на разработката не са извършени промени, засягащи планираните функционалности. Поставените в началото задачи са успешно изпълнени. Като резултат може да обобщим, че целта за създаване на уеб приложение за управление на флотилия от превозни средства, е осъществена.

7.2.Насоки за бъдещо развитие и усъвършенстване

Съществуват следните възможности за бъдещото развитие на системата:

1. **Интеграция с телематично оборудване**
2. **Дописване на *unit* тестове** – В настоящата версия са инсталирани и конфигурирани всички библиотеки за писане на *unit* тестове на сървъра и клиента. Написани са тестове на избрани файлове, но е необходимо добавянето на още с цел повишаване на процента от кода, покрит с тестове.
3. **Визуализиране на нови характеристики в модула за репорти** – В схемата на таблиците *TelematicsDatas* и *TelematicsDataHistories* са предвидени колони за съхранение на други телематични данни. Те могат да бъдат визуализирани в модула за репорти подобно на километража и нивото на горивото. Такива са текуща скорост, географска ширина и дължина, скорост на двигателя, часове на двигателя, обороти, а също и тези, които индикират за запалване на автомобила, престой, отворена врата, поставен колан, рязко спиране или ускорение, неспазване на дистанция. Освен това модулет може да се усъвършенства и чрез опция за избиране на дължината на периода, за който да се визуализира репортът (в текущата версия е една седмица).
4. **Съвместимост с *Internet Explorer*** – Настоящата версия на системата не е конфигурирана да работи с *Internet Explorer*. При отварянето и в този браузър има проблеми със стиловете на потребителския интерфейс, а също така и при зареждането на сървърните данни.
5. **Страница за преглед на компания, шофьор, превозно средство и задача** – В настоящата версия на системата в таблиците на различните страници няма икона, навигираща към страница за преглед на техните характеристики. Те могат да бъдат видени само в таблиците. При обогатяване на функционалностите (например с добавяне на коментари) тази информация няма да може да бъде поместена в таблиците. Затова ще е необходимо създаването на страници за преглед на характеристиките на компания, шофьор, превозно средство и задача.
6. **Коментари и качване на документи към компания, шофьор, превозно средство и задача** - Към формите за създаване и редактиране на компания, шофьор, превозно средство и задача могат да бъдат добавени полета за създаване на коментари и качване на документи.
7. **Разширяване на модула за превозни средства с нови функционалности** – Сега при преглед на превозно средство от навигационната лента потребителят може да избира между два таба - задачи по обслужването и репорти. В бъдеще на тази навигационна лента може да бъдат разположени още бутони към нови модули на системата:
 - **GPS проследяване** – Възможна е интеграцията с *Google Maps* карта, на която чрез използване на телематичните данни за географска ширина и дължина, да бъде проследено движението на превозното средство в реално време.

- **Инспекции** – Списъци за детайлна проверка на различни части от превозното средство;
 - **Гориво** – Този модул може да се използва за изчисляване на разходите за гориво;
- 8. История на дейностите** – Добавянето на история на дейностите в системата би било полезно за потребителите;
- 9. Управление на потребителския профил** – Сега приложението предоставя възможност единствено за визуализиране на потребителското име и имейла на вписания потребител. В бъдеще ще трябва да бъде добавена възможност за по-детайлен преглед и редактиране на информацията за потребителя и неговите идентификационни данни.

Използвана литература

- [1] Deloitte, “Fleet management in Europe”, July 2017, <https://www2.deloitte.com/content/dam/Deloitte/cz/Documents/consumer-and-industrial/cz-fleet-management-in-europe.pdf>
- [2] Chang, Jenny. “Best Fleet Management Software Reviews & Comparisons | 2019 List of Expert's Choices.” Financesonline.com, fleet-management.financesonline.com/.
- [3] “What Is The Purpose Of Fleet Management?” Financesonline.com, FinancesOnline.com, financesonline.com/what-is-the-purpose-of-fleet-management/.
- [4] “What Is Telematics? - Definition from WhatIs.com.” SearchNetworking, searchnetworking.techtarget.com/definition/telematics.
- [5] “What Is a Fleet Management Company?” CarAdvise, 6 Oct. 2018, caradvise.com/what-is-a-fleet-management-company/.
- [6] “What Is Telematics?” Verizon Connect, www.verizonconnect.com/nz/glossary/what-is-telematics/.
- [7] “What Is Telematics?” Geotab Blog, 9 Jan. 2018, www.geotab.com/blog/what-is-telematics/.
- [8] Schmidt, Marty. “Uncover All Hidden Lifecycle Ownership Costs. Find TCO in 6 Steps.” Business Case Web Site, © Solution Matrix Ltd 2004, 12 May 2018, www.business-case-analysis.com/total-cost-of-ownership.html.
- [9] “Telematics – past, present and future”, May 2008, http://www.asashop.org/wp-content/uploads/ASAtelematics_0508.pdf
- [10] “How Does Telematics Work?” TeletracNav, www.teletracnavman.com/gps-tracking-resources/fleet-management-faq/how-does-telematics-work.
- [11] “Vehicle Telematics: How Does a Vehicle Telematics Solution Work?” Embitel, 17 Dec. 2018, www.embitel.com/blog/embedded-blog/tech-behind-telematics-explained-how-does-a-vehicle-telematics-solution-work.
- [12] “The ultimate guide to fleet telematics”, Fleetcarma, <https://5vtj648dfk323byvjb7k1e9w-wpengine.netdna-ssl.com/wp-content/uploads/2016/08/ultimate-telematics-guide-pdfversion.pdf>
- [13] “What Is General Packet Radio Service (GPRS)? - Definition from Techopedia.” Techopedia.com, www.techopedia.com/definition/4473/general-packet-radio-service-gprs.
- [14] “What Is ECU (Electronic Control Unit)?” Computer Hope, 4 Oct. 2017, www.computerhope.com/jargon/e/ecu.htm.
- [15] “What Is a CAN bus?”, 23 Sep. 2017, <https://www.quora.com/What-is-a-CAN-bus>

- [16] “What Is Board Support Package? - Definition from WhatIs.com.” WhatIs.com, whatis.techtarget.com/definition/board-support-package.
- [17] Walford, Lynn. “Definition of Connected Car – What Is the Connected Car? Defined.” Auto Connected Car News, AUTO Connected Car News, 16 July 2018, www.autoconnectedcar.com/definition-of-connected-car-what-is-the-connected-car-defined/.
- [18] Association of Equipment Manufacturers. “New Mixed-Fleet Telematics Standard Earns ISO Approval.” Association of Equipment Manufacturers, 19 Aug. 2016, www.aem.org/news/july-2016/new-mixed-fleet-telematics-standard-earns-iso-approval/.
- [19] “Telematics Benefits for the Greater Good.” Geotab Blog, 5 July 2017, www.geotab.com/blog/telematics-benefits-greater-good/.
- [20] “Simple Fleet Maintenance Software for Fleet Management - AUTOsist.” Fleet Maintenance & Management Tracking Software - AUTOsist, autosist.com/.
- [21] “Compare AUTOsist vs FleetOR 2019 | FinancesOnline.” Financesonline.com, comparisons.financesonline.com/autosist-vs-fleetor.
- [22] “IMS, Fleet Maintenance Software for Vehicles and Equipment.” Innovative Maintenance Systems, www.mtcpro.com/.
- [23] “Fleetio.” Atom, Fleetio, www.fleetio.com/.
- [24] Madavu, Suchith. “CodeLog.” Requesting REST Webservice with JSON in C# Xamarin Android, 23 June 2015, www.appliedcodelog.com/2015/06/requesting-rest-webservice-with-json-in.html.
- [25] “What Is REST API Design?” MuleSoft, 20 Nov. 2017, www.mulesoft.com/resources/api/what-is-rest-api-design.
- [26] Tutorialspoint.com. “Entity Framework Code First Approach.” www.tutorialspoint.com, Tutorial Point, www.tutorialspoint.com/entity_framework/entity_framework_code_first_approach.htm.
- [27] “What Is Entity Framework?” Entity Framework Tutorial, www.entityframeworktutorial.net/what-is-entityframework.aspx.
- [28] Puhi, Maninder Singh. “Getting Started with ASP.Net Web API 2 Using CodeFirst.” CodeProject, 29 Sept. 2014, www.codeproject.com/Articles/821439/Getting-started-with-ASP-Net-Web-API-using-CodeF.
- [29] “Token Based Authentication for Web API's.” ECanarys, www.ecanarys.com/Blogs/ArticleID/308/Token-Based-Authentication-for-Web-APIs.
- [30] Trivedi, Jignesh. “Token Based Authentication Using ASP.Net Web API, OWIN and Identity With Entity Framework.”, C# Corner, www.c-

sharpcorner.com/UploadFile/ff2f08/token-based-authentication-using-Asp-Net-web-api-owin-and-i/.

[31] Lahma, Marko. “Quartz.NET - Frequently Asked Questions.” Frequently Asked Questions | Quartz.NET Documentation, www.quartz-scheduler.net/documentation/faq.html.

[32] Mahmood, Hamza. “Advantages of Developing Modern Web Apps with React.js.” Medium.com, Medium, 27 May 2018, medium.com/@hamzamahmood/advantages-of-developing-modern-web-apps-with-react-js-8504c571db71.

[33] TechMagic. “ReactJS vs Angular5 vs Vue.js - What to Choose in 2018?” Medium.com, Medium, 16 Mar. 2018, medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d.

[34] <https://npmstat.com/charts.html?package=react&package=vue&package=angular&from=2016-06-01&to=2018-05-31>

[35] Tassinari, Olivier. “Material-UI v1 Is out.” Medium.com, Medium, 17 May 2018, medium.com/material-ui/material-ui-v1-is-out-e73ce13463eb.

[36] Recharts. “Recharts/Recharts.” *GitHub*, 16 Feb. 2019, github.com/recharts/recharts.

[37] Craig, William. “Information Architecture 101: Techniques and Best Practices.” WebFX Blog, 18 Mar. 2019, www.webpagefx.com/blog/web-design/information-architecture-101-techniques-and-best-practices/

[38] “Telematics Is Revolutionising Fleet Management.” Financial Times, 18 Apr. 2016, www.ft.com/content/ca557812-c03a-11e5-9fdb-87b8d15baec2?siteedition=intl#axzz4JVuwLBje.

Приложение 1 Тестови сценарии

№	Тестови сценарий	Резултат	Статус
1.	Всички задължителни полета са означени със символа звезда (*)	Успех	Разрешен
2.	Валидация при празни задължителни полета	Успех	Разрешен
3.	Валидация на имейл, телефонен номер и потвърждение на паролата	Неуспех	Разрешен
4.	При празни задължителни полета бутонът <i>Save</i> е неактивен	Успех	Разрешен
5.	При валидация на данните името на невалидното поле и линията под него се оцветяват в червено във всички форми на системата	Успех	Разрешен
6.	Прозорци за потвърждение се визуализират при всички операции за изтриване в системата	Успех	Разрешен
7.	Проверка за правописни и граматически грешки	Успех	Разрешен
8.	Валидация при въвеждане на отрицателни числа в полетата <i>Time Rule</i> , <i>Time Reminder</i> , <i>Mileage Rule</i> и <i>Mileage Reminder</i> във формата за създаване (редактиране) на задача	Неуспех	Разрешен
9.	При многократно натискане на бутона <i>Save</i> във формата за създаване на компания, шофьор, превозно средство или задача не настъпва създаване на голям брой идентични записи	Неуспех	Разрешен

Таблица 1: „Сценарии за тестване на общи за цялата система компоненти”

№	Тестови сценарий	Резултат	Статус
1.	Всички полета във формите на системата са подравнени	Успех	Разрешен
2.	Осигурено е достатъчно място между заглавията на полетата и редовете в различните форми	Успех	Разрешен
3.	Полетата във формите на системата са озаглавени и имат изчерпателно описание	Успех	Разрешен
4.	Осигурено е достатъчно място между редовете и колоните в таблиците от системата	Успех	Разрешен
5.	Всички колони на таблиците в системата имат изчерпателно заглавие	Успех	Разрешен
6.	Бутоните в неактивно състояние са сиви и при кликуване върху тях не се случва нищо	Успех	Разрешен
7.	На потребителя не е разрешено да пише в полетата на падащите списъци	Успех	Разрешен
8.	Клавишните комбинации <i>Tab</i> и <i>Shift + Tab</i> за преминаване през елементите на формите в системата работят	Успех	Разрешен

Таблица 2: „Сценарии за тестване на потребителския интерфейс”

№	Тестови сценарий	Резултат	Статус
1.	Темата на имейла е непразна и е <i>Overdue services</i> и <i>Reminder services</i> съответно за пресрочени и предстоящи задачи	Успех	Разрешен
2.	Всички полета за заместване в шаблона на имейлите са успешно заменени с реални данни	Успех	Разрешен
3.	Изпращачът е <i>fleet.management.official@gmail.com</i>	Успех	Разрешен
4.	Имейлите изглеждат добре на различни имейл клиенти	Успех	Разрешен

Таблица 3: „Сценарии за тестване на изпратените имейли”

№	Тестови сценарий	Резултат	Статус
1.	Проверка за риск от неоторизиран достъп чрез въвеждане на <i>URL</i> -а в браузъра	Успех	Разрешен
2.	Полетата за въвеждане на пароли и тяхното потвърждение нямат функционалност за автоматично дописване	Успех	Разрешен
3.	Съдържанието на полетата за въвеждане на пароли и тяхното потвърждение не се визуализира в явен вид на екрана	Успех	Разрешен
4.	При опит за взаимодействие със системата след изтичане на оторизационния токен потребителят е пренасочен към страницата за вписване	Неуспех	Разрешен
5.	Проверка на функционалността за регистрация в системата	Успех	Разрешен
6.	Проверка на функционалността за вписване в системата	Успех	Разрешен
7.	Проверка на функционалността за излизане от системата	Успех	Разрешен

Таблица 4: „Сценарии за тестване на сигурността”

Приложение 2 Test cases

№	1.
Test case	Регистрация на потребител с валидни данни
Стъпки	<ol style="list-style-type: none"> Отваряне на системата в браузъра; Натискане на бутона <i>Register</i> от навигационната лента; Попълване на регистрационната форма със следните данни: <ul style="list-style-type: none"> Username: Maria Miller Email: mmiller@gmail.com Password: @135,_ Confirm password: @135,_ Натискане на бутона <i>Register</i>;
Очакван резултат	Пренасочване към страницата за вписване;
Резултат от изпълнението на теста	Пренасочване към страницата за вписване;
Резултат	Успех

Таблица 5: „Test case за регистрация на потребител с валидни данни”

№	2.
Test case	Регистрация на потребител при неправилно потвърждение на паролата
Стъпки	<ol style="list-style-type: none"> Отваряне на системата в браузъра; Натискане на бутона <i>Register</i> от навигационната лента; Попълване на регистрационната форма със следните данни: <ul style="list-style-type: none"> Username: Maria Miller Email: mmiller@gmail.com Password: @135,_ Confirm password: 2135,_ Натискане на бутона <i>Register</i>;
Очакван резултат	Полето за потвърждение на паролата е оцветено в червено. Визуализира се съобщение за грешка “ <i>Password doesn’t match</i> ”. Бутонът <i>Register</i> е неактивен.
Резултат от изпълнението на теста	Не се визуализират никакви индикации за грешка. Бутонът <i>Register</i> е активен.
Резултат	Неуспех

Таблица 6: „Test case за регистрация на потребител при неправилно потвърждение на паролата”

№	3.
Test case	Регистрация на потребител с използвано потребителско име
Стъпки	<p>5. Отваряне на системата в браузъра; 6. Натискане на бутона <i>Register</i> от навигационната лента; 7. Попълване на регистрационната форма със следните данни:</p> <ul style="list-style-type: none"> • Username: Maria Miller • Email: mmiller123@gmail.com • Password: \$135,_ • Confirm password: \$135,_ <p>8. Натискане на бутона <i>Register</i>;</p>
Очакван резултат	Визуализира се прозорец със съобщение за грешка “ <i>Name Maria Miller is already taken</i> ”. Бутонът <i>Register</i> е неактивен, докато прозорецът не се затвори с бутона X.
Резултат от изпълнението на теста	Не се визуализират никакви индикации за грешка. Бутонът <i>Register</i> е активен.
Резултат	Неуспех

Таблица 7: „Test case за регистрация на потребител с използвано потребителско име”

№	4.
Test case	Регистрация на потребител с използван имейл
Стъпки	<p>9. Отваряне на системата в браузъра; 10. Натискане на бутона <i>Register</i> от навигационната лента; 11. Попълване на регистрационната форма със следните данни:</p> <ul style="list-style-type: none"> • Username: Marta Miller • Email: mmiller@gmail.com • Password: \$135,_ • Confirm password: \$135,_ <p>12. Натискане на бутона <i>Register</i>;</p>
Очакван резултат	Визуализира се прозорец със съобщение за грешка “ <i>Email 'mmiller@gmail.com' is already taken.</i> ”. Бутонът <i>Register</i> е неактивен, докато прозорецът не се затвори с бутона X.
Резултат от изпълнението на теста	Не се визуализират никакви индикации за грешка. Бутонът <i>Register</i> е активен.
Резултат	Неуспех

Таблица 8: „Test case за регистрация на потребител с използван имейл”

№	5.
Test case	Вписване на потребител с валидни данни
Стъпки	<ol style="list-style-type: none"> Отваряне на системата в браузъра; Попълване на формата за вписване със следните данни: <ul style="list-style-type: none"> Username: Maria Miller Password: @135, _ Натискане на бутона <i>Login</i>;
Очакван резултат	Пренасочване към началната страница на системата;
Резултат от изпълнението на теста	Пренасочване към началната страница на системата;
Резултат	Успех

Таблица 9: „Test case за вписване на потребител с валидни данни”

№	6.
Test case	Вписване на потребител с грешно потребителско име и/или парола
Стъпки	<ol style="list-style-type: none"> Отваряне на системата в браузъра; Попълване на формата за вписване със следните данни (грешно потребителско име и парола): <ul style="list-style-type: none"> Username: Mari Miller Password: 2135, _ Натискане на бутона <i>Login</i>;
Очакван резултат	Визуализира се прозорец със съобщение за грешка “ <i>The user name or password is incorrect.</i> ”. Бутонът <i>Login</i> е неактивен, докато прозорецът не се затвори с бутона X.
Резултат от изпълнението на теста	Не се визуализират никакви индикации за грешка. Бутонът <i>Login</i> е активен.
Резултат	Неуспех

Таблица 10: „Test case за вписване на потребител с грешно потребителско име и/или парола”

№	7.
Test case	Създаване на компания с валидни данни
Предварителни данни	За изпълнението на този <i>test case</i> е необходимо в системата да бъде регистриран поне още един потребител, различен от създателя на

	компанията. За целта трябва да бъде изпълнен <i>test case №1</i> с <i>Username: George Felton</i> и <i>Email: gfelton@gmail.com</i> .
Стъпки	<ol style="list-style-type: none"> 1. Натискане на бутона <i>Create Company</i> от началната страница на системата; 2. Попълване на формата за създаване на компания със следните данни: <ul style="list-style-type: none"> • Name: Super Trans • Email: supertrans123@yahoo.com • Address: Madison, 1820 Irish Lane • Phone: 0414261200 • Allow access to users: George Felton 3. Натискане на бутона <i>Save</i>;
Очакван резултат	Пренасочване към началната страница на системата, където в списъка с компании е добавен нов ред с данните (<i>Name, Email, Address, Phone</i>) на създадената компания;
Резултат от изпълнението на теста	Пренасочване към началната страница на системата, където в списъка с компании е добавен нов ред с данните (<i>Name, Email, Address, Phone</i>) на създадената компания;
Резултат	Успех

Таблица 11: „Test case за създаване на компания с валидни данни”

№	8.
Test case	Създаване на шофьор към компания с валидни данни
Стъпки	<ol style="list-style-type: none"> 1. Избиране на иконата <i>Preview</i> срещу компанията <i>Super Trans</i>; 2. Натискане на бутона <i>Create Driver</i>; 3. Попълване на формата за създаване на шофьор със следните данни: <ul style="list-style-type: none"> • Name: James L Payton • Email: jermey_hill2@yahoo.com • Address: Scranton, 594 Dog Hill Lane • Phone: 620-561-4195 4. Натискане на бутона <i>Save</i>;
Очакван резултат	Пренасочване към страницата за преглед на шофьори, където в списъка е добавен нов ред с данните (<i>Name, Email, Address, Phone</i>) на създадения шофьор;
Резултат от изпълнението	Пренасочване към страницата за преглед на шофьори, където в списъка е добавен нов ред с данните (<i>Name, Email, Address, Phone</i>)

на теста	на създадения шофьор;
Резултат	Успех

Таблица 12: „Test case за създаване на шофьор към компания с валидни данни”

№	9.
Test case	Създаване на шофьор към компания с невалиден имейл
Стъпки	<ol style="list-style-type: none"> Избиране на иконата <i>Preview</i> срещу компанията <i>Super Trans</i>; Натискане на бутона <i>Create Driver</i>; Попълване на формата за създаване на шофьор със следните данни: <ul style="list-style-type: none"> Name: James L Payton Email: jermey_hill2 Address: Scranton, 594 Dog Hill Lane Phone: 620-561-4195 Натискане на бутона <i>Save</i>;
Очакван резултат	При напускане на полето <i>Email</i> след въвеждането на имейла се визуализира съобщение за грешка “ <i>Please enter a valid email!</i> ”. Бутонът <i>Save</i> е неактивен.
Резултат от изпълнението на теста	Не се визуализират никакви индикации за грешка. Бутонът <i>Save</i> е активен.
Резултат	Неуспех

Таблица 13: „Test case за създаване на шофьор към компания с невалиден имейл”

№	10.
Test case	Създаване на превозно средство към компания с валидни данни
Стъпки	<ol style="list-style-type: none"> Избиране на иконата <i>Preview</i> срещу компанията <i>Super Trans</i> от списъка с компании от началната страница на системата; Избиране на опцията <i>Vehicles</i> от страничното меню; Натискане на бутона <i>Create Vehicle</i>; Попълване на формата за създаване на превозно средство със следните данни: <ul style="list-style-type: none"> VIN: 3C6JR6AG8DG538799 Plate Number: 136-SMTR Type: car Brand: Toyota Model: Yaris

	<ul style="list-style-type: none"> • Select Driver: James L Payton <p>5. Натискане на бутона <i>Save</i>;</p>
Очакван резултат	Пренасочване към страницата за преглед на превозни средства, където в списъка е добавен нов ред с данните (<i>VIN, Plate number, Type, Brand, Model</i>) на създаденото превозно средство;
Резултат от изпълнението на теста	Пренасочване към страницата за преглед на превозни средства, където в списъка е добавен нов ред с данните (<i>VIN, Plate number, Type, Brand, Model</i>) на създаденото превозно средство;
Резултат	Успех

Таблица 14: „Test case за създаване на превозно средство към компания с валидни данни”

№	11.
Test case	Редактиране на превозно средство към компания
Стъпки	<ol style="list-style-type: none"> 1. Избиране на иконата <i>Preview</i> срещу компанията <i>Super Trans</i> от списъка с компании от началната страница на системата; 2. Избиране на опцията <i>Vehicles</i> от страничното меню; 3. Избиране на иконата <i>Edit</i> срещу превозното средство с <i>VIN: 3C6JR6AG8DG538799</i>, създадено с <i>test case №10</i>; 4. Редактиране на следните полета с посочените стойности: <ul style="list-style-type: none"> • Type: mini van • Model: Corolla Verso • Select driver: No Driver 5. Натискане на бутона <i>Save</i>;
Очакван резултат	Пренасочване към страницата за преглед на превозни средства, където в списъка редакциите по превозното средство са правилно отразени;
Резултат от изпълнението на теста	Пренасочване към страницата за преглед на превозни средства, където в списъка редакциите по превозното средство са правилно отразени;
Резултат	Успех

Таблица 15: „Test case за редактиране на превозно средство”

№	12.
Test case	Създаване на задача, базирана на време към превозно средство, с валидни данни
Стъпки	1. Избиране на иконата <i>Preview</i> срещу компанията <i>Super</i>

	<p><i>Trans</i> от списъка с компании от началната страница на системата;</p> <ol style="list-style-type: none"> Избиране на опцията <i>Vehicles</i> от страничното меню; Избиране на иконата <i>Preview</i> срещу превозното средство с <i>VIN: 3C6JR6AG8DG538799</i>, създадено с <i>test case №10</i>; Натискане на бутона <i>Create Service</i>; Попълване на формата за създаване на задачи със следните стойности: <ul style="list-style-type: none"> Name: Car wash Notifications Recipient: mmiller@gmail.com Description: Wash the car in the neighbourhood car wash Избиране на радио бутона Time Time Rule: 21 Time Rule Unit: Days Time Reminder: 2 Time Reminder Unit: Days Натискане на бутона <i>Save</i>;
Очакван резултат	Пренасочване към страницата за преглед на задачи, където в списъка е добавен нов ред с данните (<i>Name, Notifications Recipient, Description, Time Rule, Time Reminder</i>) на създадената задача;
Резултат от изпълнението на теста	Пренасочване към страницата за преглед на задачи, където в списъка е добавен нов ред с данните (<i>Name, Notifications Recipient, Description, Time Rule, Time Reminder</i>) на създадената задача;
Резултат	Успех

Таблица 16: „Test case за създаване на задача, базирана на време с валидни данни”

№	13.
Test case	Създаване на задача, базирана на километраж към превозно средство с валидни данни
Стъпки	<ol style="list-style-type: none"> Избиране на иконата <i>Preview</i> срещу компанията <i>Super Trans</i> от списъка с компании от началната страница на системата; Избиране на опцията <i>Vehicles</i> от страничното меню; Избиране на иконата <i>Preview</i> срещу превозното средство с <i>VIN: 3C6JR6AG8DG538799</i>, създадено с <i>test case №10</i>; Натискане на бутона <i>Create Service</i>; Попълване на формата за създаване на задачи със следните стойности: <ul style="list-style-type: none"> Name: Change the engine oil Notifications Recipient: mmiller@gmail.com

	<ul style="list-style-type: none"> • Description: Buy new oil for the change • Избиране на радио бутона Mileage • Mileage Rule: 10000 • Mileage Reminder: 500 <p>6. Натискане на бутона <i>Save</i>;</p>
Очакван резултат	Пренасочване към страницата за преглед на задачи, където в списъка е добавен нов ред с данните (<i>Name, Notifications Recipient, Description, Mileage Rule, Mileage Reminder</i>) на създадената задача;
Резултат от изпълнението на теста	Пренасочване към страницата за преглед на задачи, където в списъка е добавен нов ред с данните (<i>Name, Notifications Recipient, Description, Mileage Rule, Mileage Reminder</i>) на създадената задача;
Резултат	Успех

Таблица 17: „Test case за създаване на задача, базирана на километраж с валидни данни”

№	14.
Test case	Създаване на задача, базирана на километраж към превозно средство с отрицателна стойност за <i>Mileage Rule</i> .
Стъпки	<p>7. Избиране на иконата <i>Preview</i> срещу компанията <i>Super Trans</i> от списъка с компании от началната страница на системата;</p> <p>8. Избиране на опцията <i>Vehicles</i> от страничното меню;</p> <p>9. Избиране на иконата <i>Preview</i> срещу превозното средство с <i>VIN: 3C6JR6AG8DG538799</i>, създадено с <i>test case №10</i>;</p> <p>10. Натискане на бутона <i>Create Service</i>;</p> <p>11. Попълване на формата за създаване на задачи със следните стойности:</p> <ul style="list-style-type: none"> • Name: Change the engine oil • Notifiations Recipient: mmiller@gmail.com • Description: Buy new oil for the change • Избиране на радио бутона Mileage • Mileage Rule: -10000 • Mileage Reminder: 500 <p>12. Натискане на бутона <i>Save</i>;</p>
Очакван резултат	При напускане на полето <i>Mileage Rule</i> след въвеждането на стойността се визуализира съобщение за грешка “ <i>Please enter a positive mileage rule!</i> ”. Бутонът <i>Save</i> е неактивен.
Резултат от изпълнението на теста	Не се визуализират никакви индикации за грешка. Бутонът <i>Save</i> е активен.

Резултат	Неуспех
-----------------	---------

Таблица 18: „Test case за създаване на задача, базирана на километраж с отрицателна стойност за Mileage Rule”

№	15.
Test case	Отбелязване на задача към превозно средство като завършена
Стъпки	<p>13. Избиране на иконата <i>Preview</i> срещу компанията <i>Super Trans</i> от списъка с компании от началната страница на системата;</p> <p>14. Избиране на опцията <i>Vehicles</i> от страничното меню;</p> <p>15. Избиране на иконата <i>Preview</i> срещу превозното средство с VIN: 3C6JR6AG8DG538799, създадено с test case №10;</p> <p>16. Избиране на иконата <i>Mark as done</i> срещу задачата с име <i>Car wash</i>;</p> <p>17. Натискане на бутона <i>OK</i> от прозореца за потвърждение;</p>
Очакван резултат	Задачата е маркирана като завършена. Това означава, че ако тя е пресрочена, следващото време за изпълнение на задачата и нотификация за предстояща задача, ще бъдат преизчислени;
Резултат от изпълнението на теста	Задачата е маркирана като завършена. Това означава, че ако тя е пресрочена, следващото време за изпълнение на задачата и нотификация за предстояща задача, ще бъдат преизчислени;
Резултат	Успех

Таблица 19: „Test case за отбелязване на задача като завършена”

№	16.
Test case	Изтриване на задача към превозно средство
Стъпки	<p>1. Избиране на иконата <i>Preview</i> срещу компанията <i>Super Trans</i> от списъка с компании от началната страница на системата;</p> <p>2. Избиране на опцията <i>Vehicles</i> от страничното меню;</p> <p>3. Избиране на иконата <i>Preview</i> срещу превозното средство с VIN: 3C6JR6AG8DG538799, създадено с test case №10;</p> <p>4. Избиране на иконата <i>Delete</i> срещу задачата с име <i>Car wash</i>;</p> <p>5. Натискане на бутона <i>OK</i> от прозореца за потвърждение;</p>
Очакван резултат	Пренасочване към страницата за преглед на задачи, където от списъка е премахната задачата с име <i>Car wash</i> ;
Резултат от изпълнението на теста	Пренасочване към страницата за преглед на задачи, където от списъка е премахната задачата с име <i>Car wash</i> ;
Резултат	Успех

Таблица 20: „Test case за изтриване на задача”

№	17.
Test case	Преглед на репорт за километраж на превозно средство
Стъпки	<ol style="list-style-type: none"> 1. Избиране на иконата <i>Preview</i> срещу компанията <i>Super Trans</i> от списъка с компании от началната страница на системата; 2. Избиране на опцията <i>Vehicles</i> от страничното меню; 3. Избиране на иконата <i>Preview</i> срещу превозното средство с VIN: 3C6JR6AG8DG538799, създадено с <i>test case №10</i>; 4. Натискане на бутона <i>Reports</i> от менюто с бутони <i>Services</i> и <i>Reports</i>;
Очакван резултат	Пренасочване към страницата с репорти, където по подразбиране се визуализира графиката с едноседмичен репорт за километража;
Резултат от изпълнението на теста	Пренасочване към страницата с репорти, където по подразбиране се визуализира графиката с едноседмичен репорт за километража;
Резултат	Успех

Таблица 21: „Test case за преглед на репорт за километраж на превозно средство”

№	18.
Test case	Преглед на репорт за гориво на превозно средство
Стъпки	<ol style="list-style-type: none"> 1. Избиране на иконата <i>Preview</i> срещу компанията <i>Super Trans</i> от списъка с компании от началната страница на системата; 2. Избиране на опцията <i>Vehicles</i> от страничното меню; 3. Избиране на иконата <i>Preview</i> срещу превозното средство с VIN: 3C6JR6AG8DG538799, създадено с <i>test case №10</i>; 4. Натискане на бутона <i>Reports</i> от менюто с бутони <i>Services</i> и <i>Reports</i>; 5. Избиране на опцията <i>Fuel</i> от падащия списък, озаглавен с <i>Reports</i>;
Очакван резултат	Визуализиране на графиката с едноседмичен репорт за горивото;
Резултат от изпълнението на теста	Визуализиране на графиката с едноседмичен репорт за горивото;
Резултат	Успех

Таблица 22: „Test case за преглед на репорт за гориво на превозно средство”