Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Магистърска програма
„Софтуерни технологии”

Предмет: Обектно-ориентиран анализ и проектиране на софтуерни системи

Зимен семестър, 2024/2025 год.

Тема №15: „Taxis Management/Inquiry System“

Курсов проект

Автори:

Радослав Стоянов Велков, фак. номер 7MI3400582

Кристиан Филипов Иванов, фак. номер 6MI3400586

януари, 2025

София

Съдържание

1 Въведение

Настоящият документ представлява документация на курсов проект по предмета „Обектно-ориентиран анализ и проектиране на софтуерни системи“.

Темата на проекта – „Система за управление и справки на таксита“ (Taxis Management/Inquiry System) е актуална в контекста на ефективното управление на фирми, предоставящи транспортни услуги. В динамичната среда на транспортната индустрия фирмите, които управляват автопарк и персонал, трябва да имат постоянен достъп до актуална информация за текущото състояние на автомобилите, графиците на служителите и ефективността на маршрутите, както и разходите и приходите, свързани с дейността на фирмата. Тази информация е от решаващо значение за вземане на бързи и обосновани управленски решения, както и за оптимизация на финансите и поддържане на услуги на високо ниво за клиентите.

Фирмите за маршрутни таксита често се сблъскват с предизвикателства, свързани с проследяването на многобройни данни, относно своите коли и персонал. В един натоварен бизнес, където времето и ефективността са ключови, липсата на добре организирана и централизирана система затруднява проследяването на техническото състояние на автомобилите, графиците на ремонтите, маршрутите, както и на натовареността и квалификацията на шофьорите. Нашето приложение цели да адресира именно тези предизвикателства, като предоставя интегрирана система, която позволява ефективно управление на ресурсите и намалява риска от неефективни разходи и административни пропуски.

Системата е проектирана с помощта на езика UML, който позволява визуално и структурирано представяне на различните функционалности и взаимодействия между компонентите. UML диаграмите, изградени в средата за моделиране Visual Paradigm, позволяват да се представи пълният цикъл на работа на системата: от регистрацията и логването на потребителите, през управлението на автомобилния парк и планирането на маршрутите, до отчитането на разходите и приходите. Visual Paradigm предлага интуитивна среда за изграждане на множество диаграми, като така осигурява пълноценен и лесен за разбиране модел на системата, който може да се използва за нейното реализиране и по-нататъшно развитие.

2 Анализ на решението

2.1 Потребителски изисквания и работен процес

2.1.1 Функционални изисквания

FR.1. Системата трябва да позволява регистриране на потребители - администратори, началници, шофьори, които да имат различен достъп до функционалностите на приложението.

FR.2. Входните данни за всяко такси включват:

- марка;

- модел;
- година на производство;
- брой места за пътници;
- товароподемност;
- текущо състояние (на разположение или в ремонт).

FR.3. Системата съхранява и разполага с предефинирани маршрути в града.

FR.4. За всеки маршрут трябва да има информация за:
- възловите точки;
- дължината му в километри;
- какъв разход на гориво е необходим за едно минаване;
- брой обиколки на ден.

FR.5. Системата трябва да изисква следните входните данни за шофьор:
- лична информация (име, ЕГН);
- професионален опит в години;
- месечна заплата в лева;
- категория на шофьорска книжка;
- текущ маршрут, който изпълнява.

FR.6. Системата трябва да може да записва и актуализира данни за всеки автомобил, като актуалното състояние (на разположение или в ремонт); времето, за което ще бъде извън строя. Всяка промяна се записва автоматично в базата данни.

FR.7. Системата трябва да поддържа информация за активните маршрути и съответните им разпределени шофьори, във всеки един момент от времето, което автоматично да се отразява в графиците на водачите.

FR.8. Системата трябва да може да генерират отчети за общите разходи и приходи на месечна база по всички възможни разбивки и филтрирания - по шофьори, по маршрути, по автомобили.

FR.9. Трябва да се поддържа актуализация на базата данни при действията:
- добавяне на нов потребител;
- добавяне на нов маршрут;
- добавяне на нов автомобил;
- актуализиране на потребителски данни;
- актуализиране на данни за маршрут;
- актуализиране на данни за автомобил;
- изтриване на каквито и да е данни.

FR.10. Системата трябва да генерира отчети за състоянието на автомобилите, включително тези в ремонт, с информация за времето за ремонт и разходите, на седмична, месечна и годишна база.

FR.11. Системата трябва да генерира отчети за маршрутите, включващи разходите и ефективността (разходи на километър, средно време за маршрут), както и отчет за назначения на шофьорите.

FR.12. Системата трябва да генерира месечен отчет с обобщение на всички разходи и приходи за управлението на фирмата, който ще се използва за планиране и оптимизация.

2.1.2 Нефункционални изисквания

NFR.1. (Сигурност) Системата трябва да осигурява криптирано съхранение на данни, включително информация за потребители, автомобили и маршрути, с контрол на достъпа, базиран на потребителските роли, както и логване чрез пароли, отговарящи на строга политика за сложност.

NFR.2. (Отказоустойчивост) Трябва да се поддържат механизми за автоматично архивиране и възстановяване на данни в случай на срив, като при временни откази данните да се запазват временно в кеш и да се синхронизират след възстановяване на системата.

NFR.3. (Производителност) Системата трябва да осигурява обработка на данни в реално време, позволявайки актуализация на графици и маршрути без забавяне, като генерирането на отчети да отнема до 2 секунди за стандартни и до 4 секунди за сложни заявки.

NFR.4. (Преносимост) Системата трябва да бъде съвместима с основните операционни системи (Windows, Linux, macOS), като интерфейсът ѝ да бъде достъпен от различни устройства, включително настолни компютри, таблети и смартфони.

NFR.5. (Изменяемост) Архитектурата трябва да позволява лесно разширяване с нови функционалности като добавяне на отчети или интеграции, чрез модули слабо свързани помежду си и с висока степен на капсулираност.

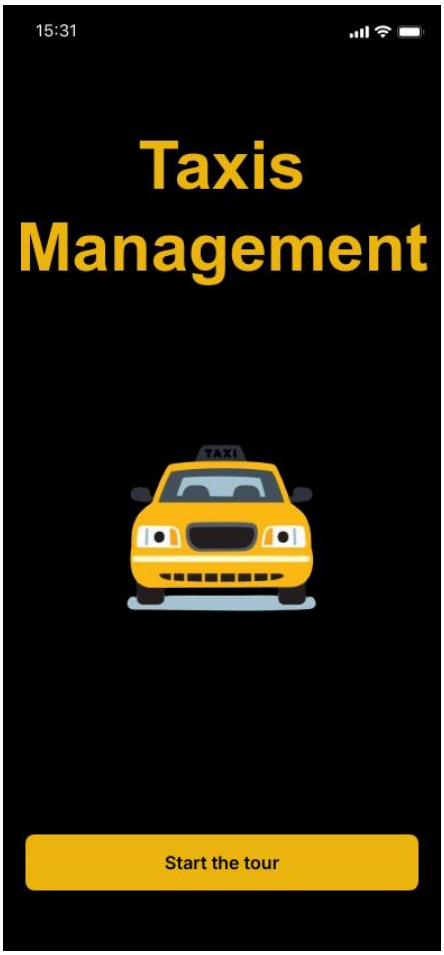
NFR.6. (Надеждност) Системата трябва да осигурява минимум 99.9% време на активност на месец, като всички критични операции, включително актуализацията на данни и генерирането на отчети, трябва да бъдат автоматично валидирани за избягване на грешки.

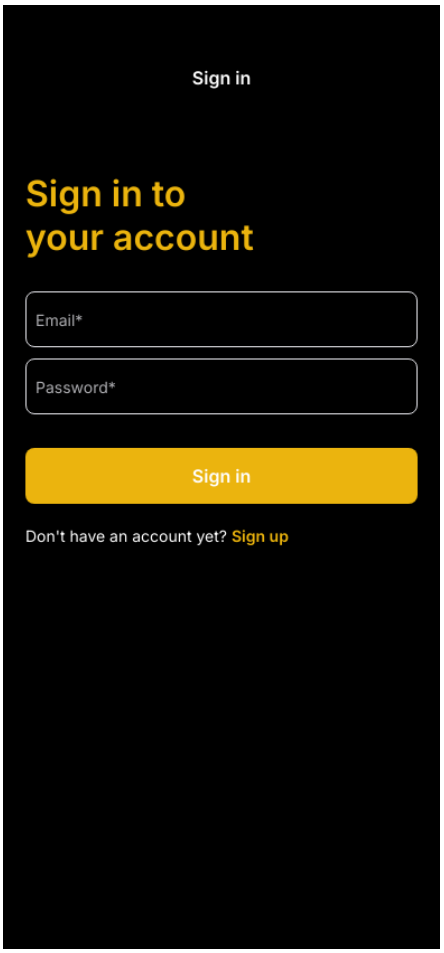
NFR.7. (Употребяемост) Потребителският интерфейс трябва да бъде интуитивен, с ясни указания, лесни за изпълнение стъпки и логическа последователност в навигацията, като метрика за това би била - обучението на нови потребители да не отнема повече от час.

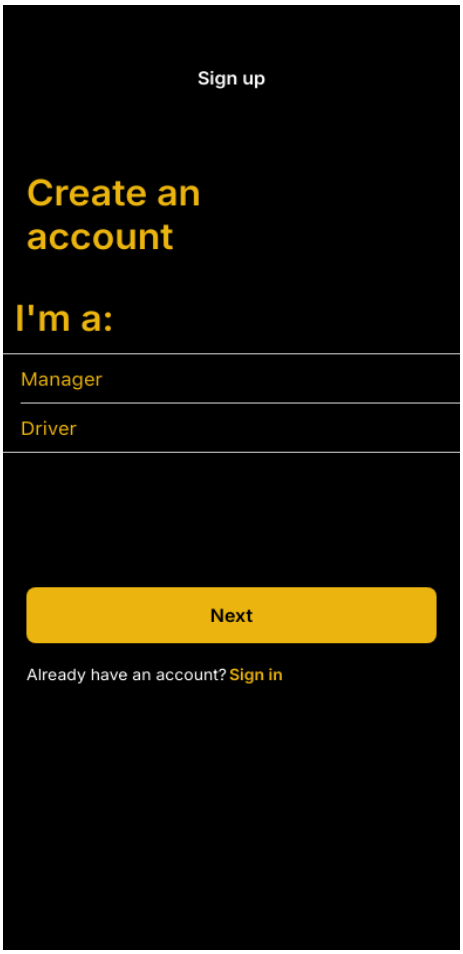
NFR.8. (Мащабируемост) Системата трябва да поддържа едновременна работа на поне 100 потребители без забавяне, като базата данни да е проектирана за обработка на големи обеми от данни, включително хиляди записи за автомобили, маршрути и отчети.

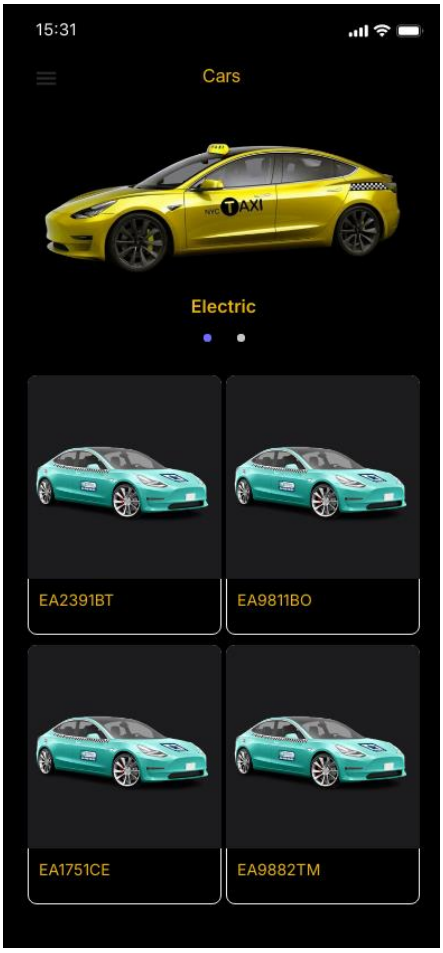
NFR.9. (Достъпност) Приложението трябва да бъде достъпно 24/7 чрез уеб интерфейс, съвместим с различни браузъри (Chrome, Firefox, Edge), като дизайнът му да бъде съобразен с нуждите на потребители с увреждания.

2.2 Примерен графичен интерфейс

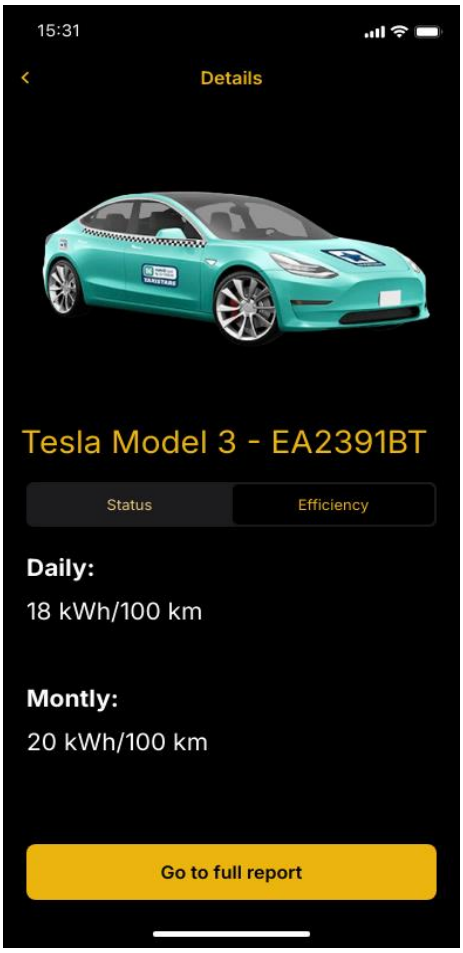


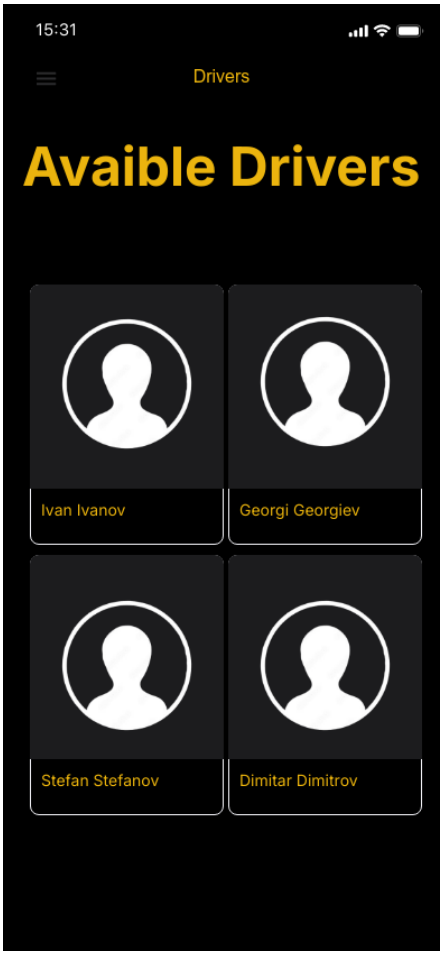


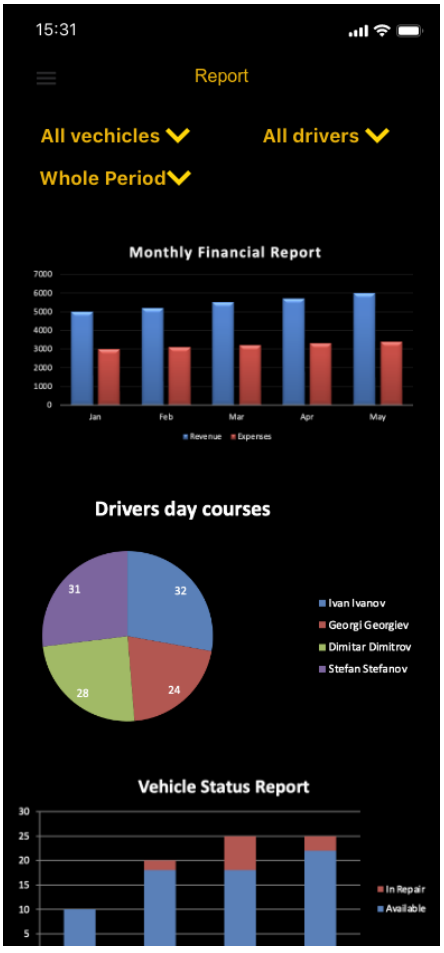



15:31
Cars
Electric
EA2391BT
EA9811BO
EA1751CE
EA9882TM


15:31
Details
Tesla Model 3 - EA2391BT
Status
Efficiency
Brand: Tesla
Model: Model 3
Year of manufacture: 2017
Seating capacity: 5
Payload capacity: 425 kg
Status: Available
Go to full report

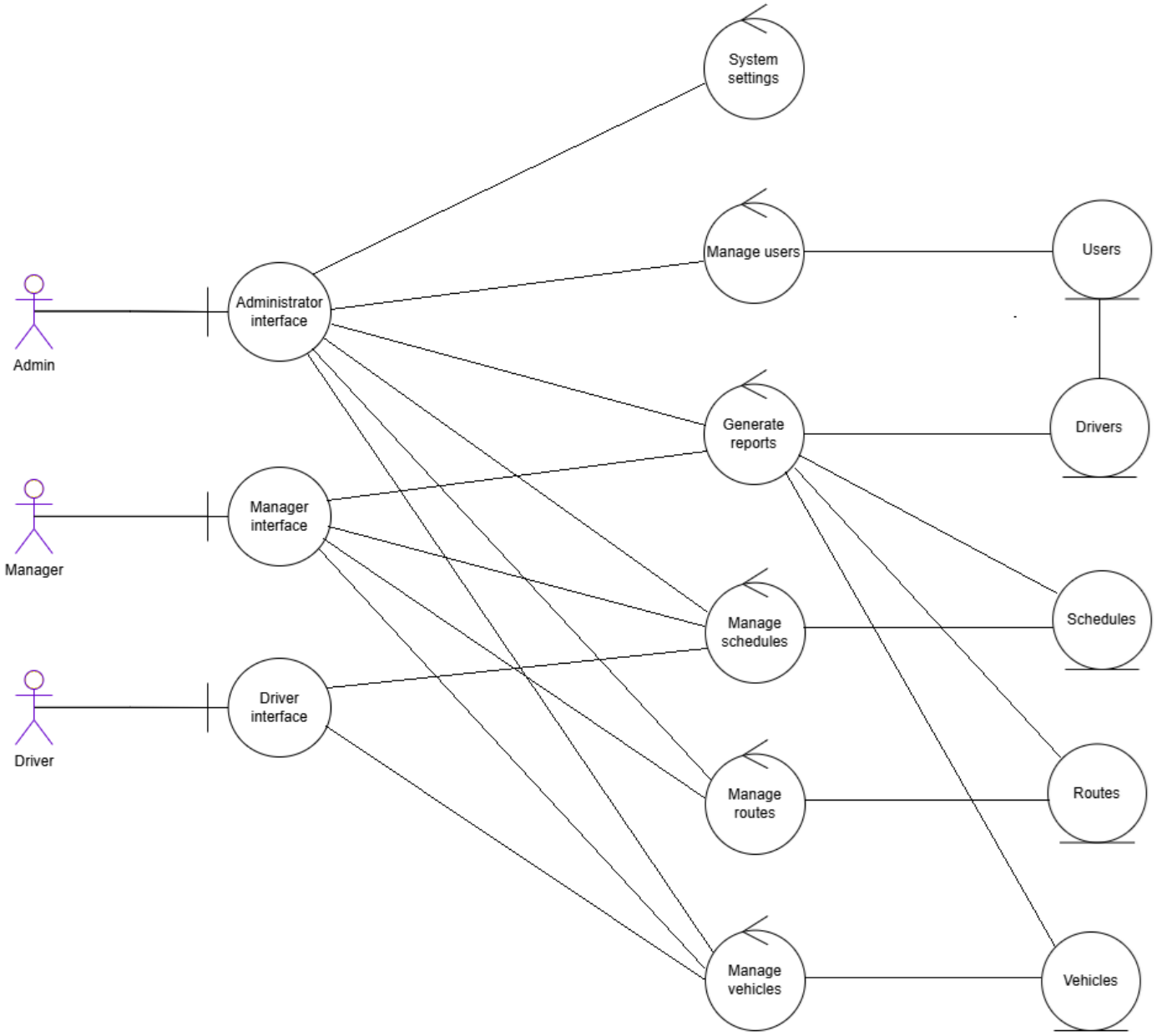

15:31
Details
Tesla Model 3 - EA2391BT
Status
Efficiency
Daily:
18 kWh/100 km
Montly:
20 kWh/100 km
Go to full report


15:31
Drivers
Avaible Drivers
Ivan Ivanov
Georgi Georgiev
Stefan Stefanov
Dimitar Dimitrov


15:31
Report
All vechicles
All drivers
Whole Period
Monthly Financial Report
Jan
Feb
Mar
Apr
May
Revenue
Expenses
Drivers day courses
Ivan Ivanov
Georgi Georgiev
Dimitar Dimitrov
Stefan Stefanov
Vehicle Status Report
In Repair
Available

2.3 Диаграми на анализа

2.3.1 Entity-Control-Boundary Pattern



Тази диаграма и описание предоставят ясна представа за архитектурата на система, като дефинират различните роли на потребителите и техните взаимодействия със системата. Реализирано е разграничение на ролите и отговорностите, като по този начин се гарантира сигурността и правилното функциониране на процесите.

1. Actors:

- **Admin:** Администраторът има пълен достъп до всички функционалности на системата, включително управление на потребители, конфигуриране на системни настройки и генериране на отчети. Той отговаря за основната администрация на системата и може да извършва всички операции, свързани с настройките, потребителите и управлението на данни. Примерни действия включват добавяне и премахване на потребители, настройване на фирмени политики, генериране на отчети за ефективността и състоянието на ресурси, както и управление на графици, маршрути и превозни средства.
- **Manager:** Началникът има ограничен достъп в сравнение с администратора, но може да управлява и актуализира работни графици, маршрути и превозни средства. Той отговаря за ежедневното управление на оперативните задачи и мониторинга на изпълнението на задачите от страна на шофьорите. Началникът може да генерира отчети за различни аспекти на дейността, като ефективност на маршрутите и техническо състояние на превозните средства. Примерни действия включват актуализиране на графици, управление на маршрути, възлагане на задачи на шофьори и генериране на отчети.
- **Driver:** Шофьорът има достъп до функционалности, които му позволяват само да преглежда своите графици и маршрути, както и да управлява състоянието на превозните средства, с които работи. Той не може да променя основните параметри на системата като маршрути или графици, но може да докладва за технически проблеми със своето превозно средство. Примерни действия включват преглеждане на работен график, докладване на проблеми със състоянието на превозни средства и изпълнение на задачи по поддръжка на превозни средства.

2. Boundaries:

- **Administrator Interface:** Това е интерфейсът, който предоставя пълен достъп до функционалностите, които са предназначени за администратора. Чрез него администраторът може да управлява всички основни елементи на системата (потребители, маршрути, графици, отчети и др.).
- **Manager Interface:** Този интерфейс предоставя достъп до функционалности за оперативно управление на маршрутите, графиците и превозните средства. Началникът може да използва интерфейса за актуализиране на графици и маршрути, както и за генериране на отчети, свързани с ефективността и използваемостта на ресурсите.
- **Driver Interface:** Интерфейсът на шофьора е предназначен за преглед на графиците и превозните средства. Шофьорът има ограничен достъп само до тези функционалности, които са необходими за изпълнение на работата му. Не може да променя маршрути или да прави други промени в системата.

3. Controllers:

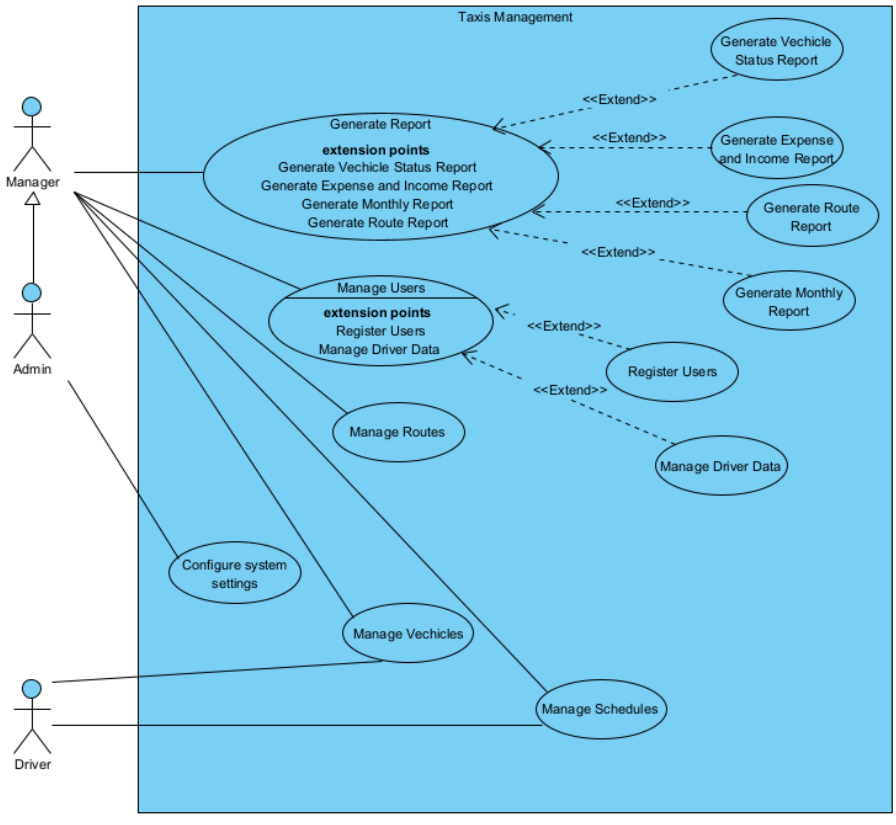
- **System Settings:** Контролер, който се свързва с интерфейса на администратора и позволява конфигуриране на основни параметри на системата. Този контролер се грижи за настройките на фирмените политики, правила за отчитане и други параметри, които могат да бъдат променяни само от администратора.

- **Manage Users:** Администраторски контролер за управление на потребители, който осигурява интерфейс за създаване, редактиране и премахване на потребители.
- **Generate Reports:** Използва се от администратор и мениджър за генериране на отчети за фирмените дейности. Включва отчети за разходи, приходи, ефективност на маршрутите, състояние на превозните средства и други. Различни типове отчети могат да бъдат генерирани от различни актьори с различни нива на достъп.
- **Manage Schedules:** Управлява всички процеси, свързани с графиците на шофьорите. Началникът може да използва този контролер за актуализиране на графици, възлагане на маршрути на шофьори и извършване на промени в работния процес.
- **Manage Routes :** Контролер за управление на маршрутите, който е достъпен за началника. Той позволява създаването, редактирането и премахването на маршрути, както и свързването им с конкретни шофьори.
- **Manage Vehicles:** Контролер, който се използва за управление на превозните средства. Той позволява добавяне, премахване и актуализиране на информация за превозните средства в системата (например, техническо състояние, статус на ремонта и др.).

4. Entities:

- **Users:** Съдържа информация за всички потребители в системата, като техните данни за вход, роля и ниво на достъп.
- **Drivers:** Съдържа информация за шофьорите - име, професионален опит, категория на шофьорската книжка и текущо изпълнявани маршрути.
- **Schedules:** Съхранява данни за работния график на шофьорите, включително работни часове и възложени маршрути.
- **Routes:** Съдържа информация за маршрути, като точки на маршрута, разходи на гориво и други свързани параметри.
- **Vehicles:** Пази данни за превозните средства - техните технически характеристики, текущо състояние и история на ремонтите.

2.3.2 UML Use Case Diagram



Описание на потребителските случаи:

Use Case ID	Actor/s	Use Case
UC-1	Manager	Generate Reports
UC-2	Manager	Manage Users
UC-3	Manager	Manage Routes
UC-4	Manager/Driver	Manage Vehicles
UC-5	Manager/Driver	Manage Schedules
UC-6	Admin	Configure System Settings

Use Case ID	1
Use Case	Генериране на отчети
Actors	Мениджър
Description	Системата позволява на потребителите да генерират различни видове отчети, включително за състоянието на автомобилите, маршрути, разходи и приходи. Отчетите се използват за анализ и оптимизация на дейността на компанията.
Trigger	Потребителят избира функция за генериране на отчет от менюто на системата.
Preconditions	1. Потребителят трябва да е регистриран и да има подходящи права за достъп. 2. Данните в базата трябва да бъдат актуални и коректно записани.
Postconditions	Success guarantees: Отчетът е успешно генериран и визуализиран на екрана или изтеглен във файл. Minimal guarantees: Потребителят получава известие за грешка, ако отчетът не може да бъде генериран.
Normal Flow	1. Потребителят влиза в системата. 2. Избира „Генериране на отчети“ от менюто. 3. Системата показва наличните типове отчети. 4. Потребителят избира тип отчет и задава филтри (ако е приложимо). 5. Системата генерира отчета и го визуализира или предоставя за изтегляне.
Alternative Flows	1. Системата информира потребителя, че липсват данни за

	генериране на избрания отчет. 2. Потребителят може да избере друг тип отчет или да въведе липсващите данни.
Exceptions	Няма интернет връзка – системата информира потребителя за проблема и препоръчва повторен опит по-късно.
Includes	Изтегляне на отчет.
Frequency of Use	Ежедневно, седмично, месечно или при поискване.
Special Requirements	1. Системата трябва да гарантира защитата на данните според изискванията на GDPR. 2. Отчетите трябва да бъдат във формат, който позволява лесно споделяне и анализ (например PDF или Excel).
Assumptions	Потребителите имат основни познания за работа със системата.
Notes and Issues	Възможно е в бъдеще да се добавят нови типове отчети според нуждите на потребителите.

Use Case ID	2
Use Case	Управление на потребители
Actors	Мениджър
Description	Системата позволява на мениджърите да добавят, актуализират и изтриват потребители, както и да задават техните права за достъп до функционалностите на приложението.
Trigger	Мениджърът избира функция за управление на потребители от менюто на системата.
Preconditions	1. Трябва регистрация и подходящи права за достъп. 2. Системата трябва да има достъп до базата данни.
Postconditions	Success guarantees: Потребителите са успешно добавени, актуализирани или изтрити, а информацията е записана в базата данни. Minimal guarantees: Системата информира мениджъра за грешка, ако операцията не може да бъде изпълнена.
Normal Flow	1. Мениджърът влиза в системата. 2. Избира „Управление на потребители“ от менюто. 3. Системата показва списък с текущи потребители. 4. Администраторът избира действе: добавяне, актуализиране или изтриване на потребител. 5. Системата извършва избраната операция и актуализира базата данни.

Alternative Flows	1. Системата информира потребителя, че няма права за извършване на операцията. 2. Потребителят може да се свърже със системен администратор за съдействие.
Exceptions	Проблем с базата данни – системата информира администратора за проблема и препоръчва повторен опит по-късно.
Includes	Няма.
Frequency of Use	При необходимост от управление на потребителски профили.
Special Requirements	1. Системата трябва да записва всички промени в лог файл за проследимост. 2. Данните на потребителите трябва да се съхраняват според изискванията на GDPR.
Assumptions	Технически умения за работа със системата.
Notes and Issues	Възможно е в бъдеще да се добавят нови роли и права за достъп.

Use Case ID	3
Use Case	Управление на маршрути
Actors	Мениджър
Description	Системата позволява на мениджърите да добавят, актуализират и изтриват маршрути, както и да управляват свързаната с тях информация, като възлови точки, дължина, разход на гориво и график на обиколки.
Trigger	Мениджърът избира функция за управление на маршрути от главното меню на системата.
Preconditions	1. Мениджърът трябва да е влязъл в системата с подходящи права за достъп. 2. Системата трябва да има достъп до базата данни.
Postconditions	Success guarantees: Данните за маршрутите са успешно добавени, актуализирани или изтрити, а промените са записани в базата данни. Minimal guarantees: Системата информира мениджъра за грешка, ако операцията не може да бъде извършена.
Normal Flow	1. Мениджърът влиза в системата. 2. Избира „Управление на маршрути“ от менюто. 3. Системата показва списък с текущи маршрути. 4. Мениджърът избира действие: добавяне на нов маршрут, актуализиране на съществуващ маршрут или изтриване на маршрут.

	5. Въвежда или редактира необходимите данни (възлови точки, дължина, разход на гориво и т.н.). 6. Системата записва промените в базата данни.
Alternative Flows	1. Системата информира потребителя, че няма права за извършване на операцията. 2. Потребителят може да се свърже със системен администратор за съдействие.
Exceptions	Проблем с базата данни – системата информира мениджъра за проблема и препоръчва повторен опит по-късно.
Includes	Няма.
Frequency of Use	При всяка необходимост от добавяне или актуализиране на маршрути.
Special Requirements	1. Системата трябва да осигурява визуализация на маршрута чрез карта. 2. Данните за маршрутите трябва да се съхраняват според изискванията за точност и консистентност.
Assumptions	Мениджърите имат достатъчно информация за съществуващите маршрути и логистичните нужди.
Notes and Issues	В бъдеще може да се наложи интеграция с външни навигационни системи.

Use Case ID	4
Use Case	Управление на превозни средства
Actors	Мениджър, Шофьор
Description	Системата позволява на мениджърите и шофьорите да добавят, актуализират и преглеждат информация за превозни средства, включително текущото им състояние (на разположение или в ремонт), товароподемност, брой места и други параметри.
Trigger	Мениджърът или шофьорът избира функция за управление на превозни средства от главното меню на системата.
Preconditions	1. Актьорът трябва да е влязъл в системата с подходящи права за достъп. 2. Системата трябва да има достъп до базата данни.
Postconditions	Success guarantees: Данните за превозните средства са успешно добавени, актуализирани или прегледани, а промените са записани в базата данни. Minimal guarantees: Системата информира актьора за грешка, ако

	операцията не може да бъде извършена.
Normal Flow	1. Актьорът влиза в системата. 2. Избира „Управление на превозни средства“ от менюто. 3. Системата показва списък с текущите превозни средства. 4. Актьорът избира действие: добавяне на ново превозно средство, актуализиране на информация или преглед на детайли за превозно средство. 5. Въвежда или редактира необходимите данни (напр. състояние, товароподемност, ремонт). 6. Системата записва промените в базата данни.
Alternative Flows	1. Системата информира потребителя, че няма права за извършване на операцията. 2. Потребителят може да се свърже със системен администратор за съдействие.
Exceptions	Проблем с базата данни – системата информира актьора за проблема и препоръчва повторен опит по-късно.
Includes	Няма.
Frequency of Use	При всяка необходимост от актуализация или преглед на данни за превозни средства.
Special Requirements	1. Системата трябва да показва статуса на превозното средство в реално време (на разположение, в ремонт). 2. Промените трябва да се отразяват незабавно в базата данни.
Assumptions	Актьорите имат достъп до точна информация за превозните средства.
Notes and Issues	Може да се наложи бъдеща интеграция със системи за GPS проследяване на превозните средства.

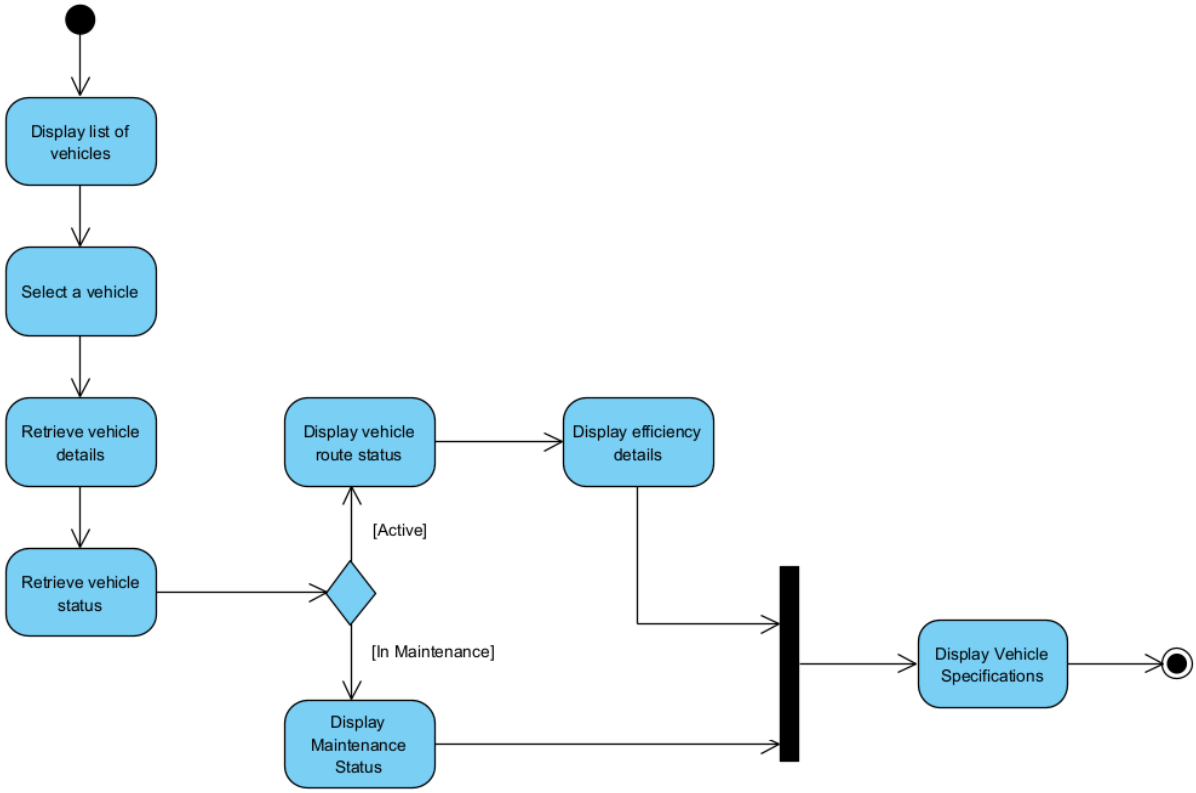
Use Case ID	5
Use Case	Управление на графици
Actors	Мениджър, Шофьор
Description	Системата позволява на мениджърите и шофьорите да управляват графици, като добавят, актуализират или премахват/променят графици и (мениджърите) разпределят шофьори за съответни маршрути.
Trigger	Потребителят избира функция за управление на графици от главното меню.
Preconditions	1. Потребителят трябва да е влязъл в системата със съответни права

	за достъп. 2. Системата трябва да има актуална информация за маршрутите и шофьорите.
Postconditions	Success guarantees: Графикът е успешно добавен, актуализиран или премахнат, а промените са записани в базата данни. Minimal guarantees: Системата информира за грешка, ако операцията не може да бъде извършена.
Normal Flow	1. Потребителят влиза в системата. 2. Избира „Управление на графици“ от менюто. 3. Системата показва наличните графици. 4. Потребителят избира действие: добавяне, актуализиране или премахване на график. 5. Въвежда или редактира необходимите данни (например разпределение на шофьори и маршрути). 6. Системата записва промените в базата данни.
Alternative Flows	1. Системата информира потребителя, че няма права за извършване на операцията. 2. Потребителят може да се свърже със системен администратор за съдействие.
Exceptions	Проблем с базата данни – системата информира за проблема и препоръчва повторен опит по-късно.
Includes	Няма.
Frequency of Use	При всяка необходимост от актуализация или преглед на графиците.
Special Requirements	1. Системата трябва да осигури визуализация на графиците в календарен формат. 2. Данните за графиците трябва да се съхраняват според изискванията за точност и консистентност.
Assumptions	Потребителите имат предходна информация за разпределението на шофьорите и маршрутите и какво точно е необходимо.
Notes and Issues	Може да се наложи интерация с външни системи за планиране на графици.

Use Case ID	6
Use Case	Конфигуриране на системни настройки
Actors	Администратор
Description	Системата позволява на администраторите да конфигурират системни настройки, като права за достъп, настройка на работни часове, настройки на отчети и други.

Trigger	Администраторът избира функция за конфигуриране на системни настройки от менюто на системата.
Preconditions	1. Администраторът трябва да е влязъл в системата с подходящи права за достъп. 2. Системата трябва да има достъп до съществуващите настройки.
Postconditions	Success guarantees: Системните настройки са успешно конфигурирани и записани в базата данни. Minimal guarantees: Системата информира за грешка, ако операцията не може да бъде извършена.
Normal Flow	1. Администраторът влиза в системата. 2. Избира „Конфигуриране на системни настройки“ от менюто. 3. Системата показва наличните настройки. 4. Администраторът избира настройка за конфигуриране и прави необходимите промени. 5. Системата записва промените в базата данни.
Alternative Flows	1. Системата информира администратора, че няма права за извършване на операцията. 2. Администраторът може да се свърже с висшестоящ мениджър за съдействие.
Exceptions	Проблем с базата данни – системата информира за проблема и препоръчва повторен опит по-късно.
Includes	Няма.
Frequency of Use	При всяка необходимост от промяна на системни настройки.
Special Requirements	1. Системата трябва да гарантира защита на данните при настройване на права за достъп и конфиденциалност.
Assumptions	Администраторът има технически познания за конфигуриране на системата и е наясно с изискванията.
Notes and Issues	Възможно е да се добавят нови настройки според нуждите на компанията.

2.3.3 UML Activity Diagram

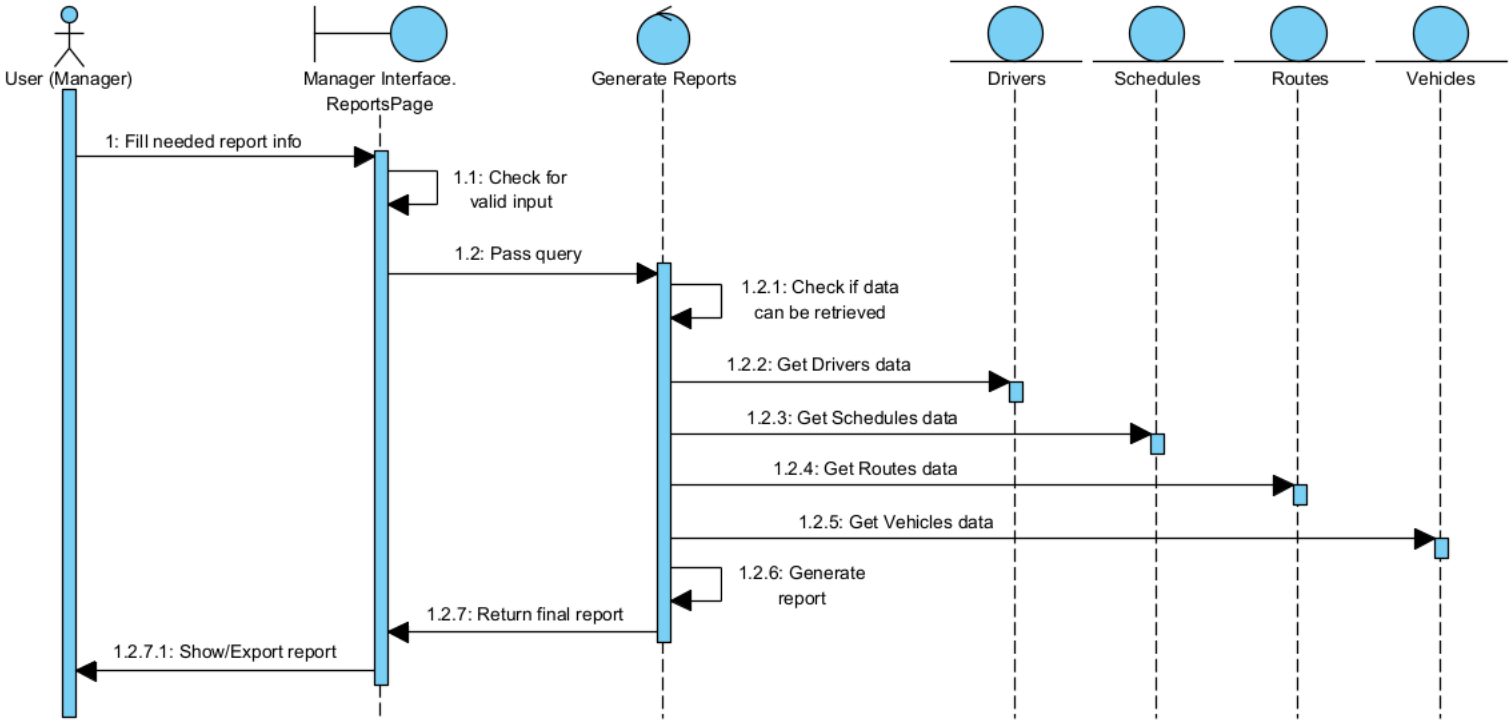


Диаграмата активност представлява процеса на обработка на информация за превозни средства. Подробно описание на всяка част от диаграмата:

1. Действие: „**Показване на списък с превозни средства**“. Това е първото действие в процеса. Системата показва списък с наличните превозни средства на потребителя.
2. Действие: „**Избор на превозно средство**“. Потребителят избира конкретно превозно средство от списъка.
3. Действие: „**Извличане на данни за превозното средство**“. След избора, системата извлича основна информация за превозното средство, като регистрационен номер, тип, модел и данни за разходите на гориво.
4. Действие: „**Извличане на статус на превозното средство**“. Системата проверява текущото състояние на превозното средство. Статусът може да бъде активно (в движение или готово за използване) или в поддръжка (недостъпно поради текущи ремонтни дейности).
5. Условно разклонение: Проверка на статус. Системата проверява дали статусът на превозното средство е активно или в поддръжка. Ако е активно, процесът преминава към показване на състояние на превозното средство. Ако е в поддръжка, процесът преминава към показване на статус на поддръжка.

6. Действие: „**Показване на състояние на маршрута на превозното средство**“. Ако превозното средство е активно, системата показва текущия или последния използван маршрут.
7. Действие: „**Показване на ефективност на превозното средство**“. Допълнително действие, което предоставя подробности за ефективността на превозното средство (разход на горивою).
8. Действие: „**Показване на статус на поддръжка**“. Ако превозното средство е в поддръжка, системата информира потребителя за текущия статус, като вид на ремонта, очаквано време за приключване, причина за поддръжката.
9. Обединение. Обединителната линия символизира, че независимо от статуса на превозното средство (активно или в поддръжка), процесът продължава към следващата стъпка.
10. Действие: „**Показване на спецификации на превозното средство**“. На последния етап системата показва пълни спецификации на превозното средство. Това включва технически характеристики.

2.3.4 UML Sequence Diagram



Диаграмата на последователността и на комуникацията илюстрира процеса на генериране на отчети от ролята Manager (Началник) чрез взаимодействие с интерфейса и системните компоненти. Тя се състои от:

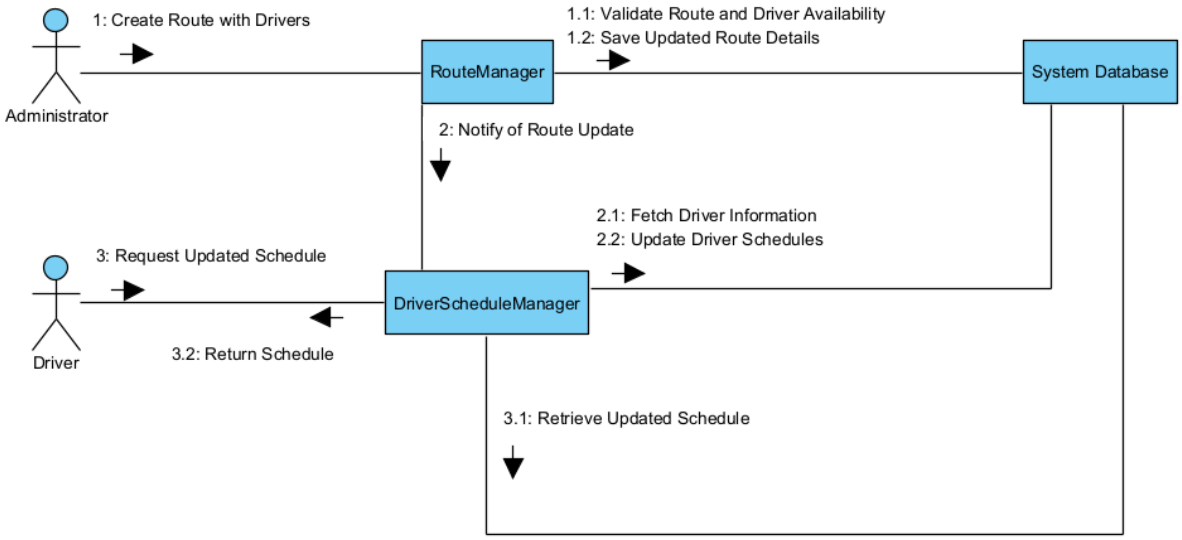
Основни компоненти:

1. **User (Manager)** - Началникът, който инициира процеса чрез попълване на необходимата информация за отчета.
2. **Manager Interface:ReportsPage** - Графичният интерфейс за управителя, който приема данните и валидира входната информация.
3. **Generate Reports** - Контролният клас, който обработва заявката за генериране на отчет и координира достъпа до различни данни.
4. **Entities (Drivers, Schedules, Routes, Vehicles)** - Информационните единици, от които се извличат данните за създаване на отчета.

Последователност от действия:

1. **Попълване на информация за отчет:** Мениджърът попълва необходимите данни в интерфейса (стъпка *1*).
2. **Валидиране на входните данни:** Интерфейсът проверява коректността на въведената информация (*стъпка 1.1*).
3. **Изпращане на заявката:** Интерфейсът предава заявката за генериране на отчета към контролера “Generate Reports” (*стъпка 1.2*).
4. **Проверка за наличност на данни:** Контролният клас проверява дали нужните данни могат да бъдат извлечени от съответните единици (*стъпка 1.2.1*).
5. **Извличане на информация: “**Generate Reports” извлича данните за: шофьорите (*стъпка 1.2.2*); графиците (*стъпка 1.2.3*); маршрутите (*стъпка 1.2.4*); автомобилите (*стъпка 1.2.5*).
6. **Генериране на отчета:** Контролерът обработва извлечените данни и създава крайния отчет (*стъпка 1.2.6*).
7. **Връщане на отчета:** Генерираният отчет се предава обратно към интерфейса, който го визуализира на потребителя с опция за експортиране (*стъпка 1.2.7*).

2.3.5 UML Communication Diagram



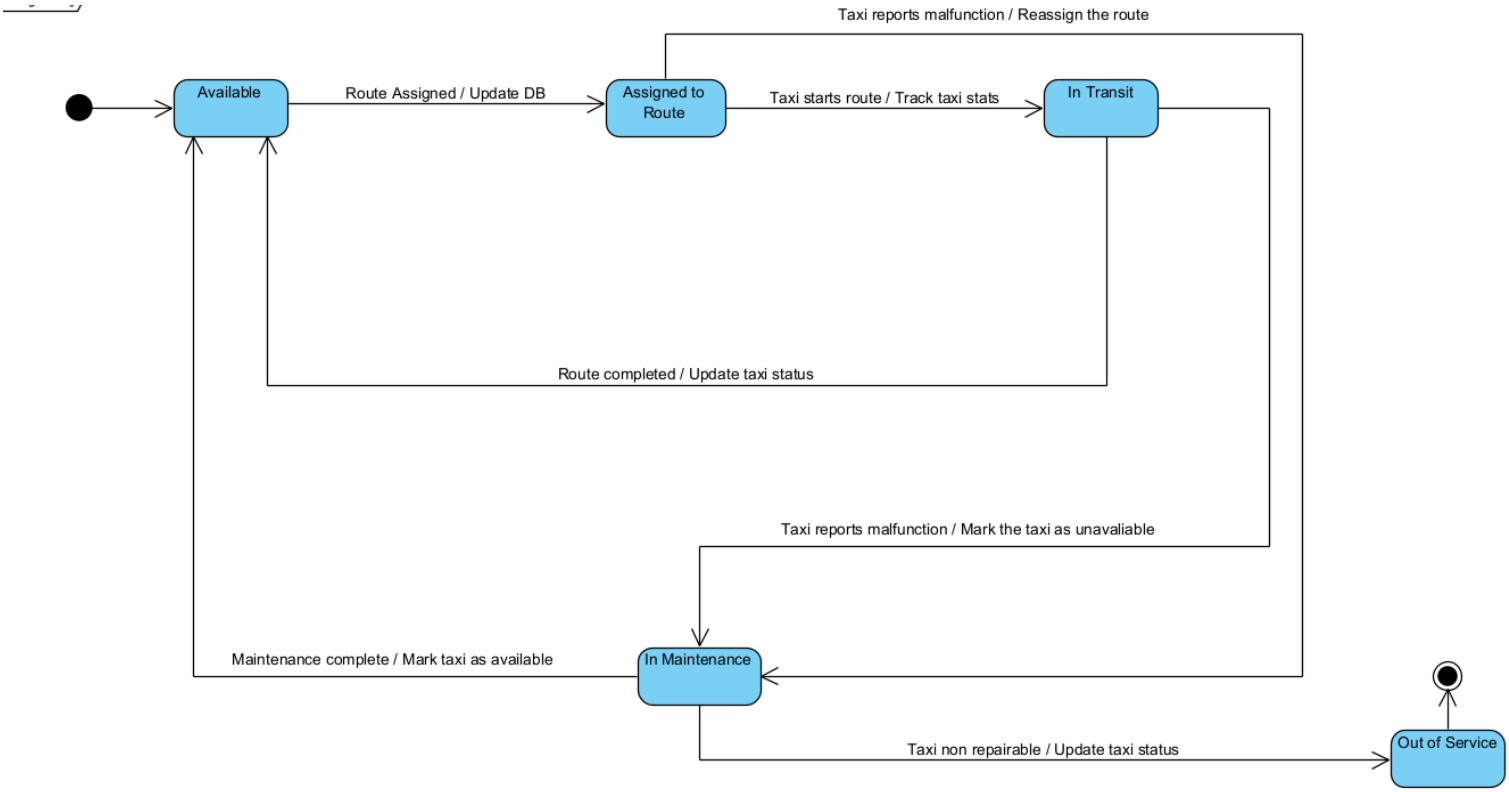
Диаграмата представлява комуникационния процес между участниците и компонентите на системата за управление на маршрути и графици на шофьори.

1. **Администраторът**:
- Инициира процеса чрез създаване или актуализиране на маршрут с разпределени шофьори (Съобщение 1).
- Тази заявка се обработва от компонента **RouteManager**, който:
    - Проверява валидността на маршрута и наличността на шофьорите (Съобщение 1.1).
    - Съхранява актуализираните данни за маршрута в базата данни (Съобщение 1.2).

2. **RouteManager**:
- След успешно създаване или актуализиране на маршрута, уведомява компонента **DriverScheduleManager** за промяната (Съобщение 2).
- **DriverScheduleManager**:
    - Извлича информация за съответните шофьори от базата данни (Съобщение 2.1).
    - Актуализира техните графици в базата данни (Съобщение 2.2).

3. **Шофьорът**:
- Може да заяви информация за актуалния си график чрез съобщение към **DriverScheduleManager** (Съобщение 3).
- **DriverScheduleManager**:
    - Извлича съответната информация за графика от базата данни (Съобщение 3.1).
    - Връща актуализирания график към шофьора (Съобщение 3.2).

2.3.6 UML State Machine Diagram



Диаграмата на състояние на машина представя жизнения цикъл и преходите на състоянието на такси в системата. Таксито може да преминава между различни състояния въз основа на дейността си. Пълно описание на структурата:

1. **Налично** - Това е началното състояние на таксито, което показва, че то е свободно и готово за възлагане на маршрут. Таксито преминава в това състояние от състоянието "В ремонт", когато поддръжката е приключила и таксито е отново готово за употреба. Също така, таксито може да се върне в това състояние, когато приключи маршрутът му, ако няма други проблеми.
2. **Възложено на маршрут** - Това състояние показва, че на таксито е възложен конкретен маршрут. В този етап таксито е планирано да започне транспортиране на пътници или товари. От това състояние таксито може да премине в "В движение", когато започне маршрута, или да се върне в "Налично", ако маршрутът е завършен успешно. Ако обаче има технически проблем, то преминава в "В ремонт".
3. **В движение** - Това състояние показва, че таксито изпълнява възложения маршрут. Системата може да следи движението и статистиките на таксито. Таксито може да се върне в "Налично", когато маршрутът е завършен, или да премине в "В ремонт", ако бъде докладван технически проблем.
4. **В ремонт** - В това състояние таксито е временно извън услуга поради необходимост от поддръжка. След приключване на ремонта то може да бъде върнато в състояние "Налично", ако е поправено успешно. Ако обаче бъде установено, че ремонтът не е възможен, таксито преминава в състояние "Извън услуга".
5. **Извън услуга** - Това е крайната точка на жизнения цикъл на таксито. В това състояние то е извадено от системата, тъй като не подлежи на ремонт или повече не е годно за експлоатация.

Преходи между състояния:

1. **Възлагане на маршрут / Актуализиране на базата данни** - Това събитие активира прехода от състояние "Налично" към "Възложено на маршрут". При възлагането на маршрут системата обновява информацията за статуса на таксито.
2. **Таксито започва маршрут / Проследяване на статистиките** - Този преход води от "Възложено на маршрут" към "В движение". Това показва, че таксито е започнало маршрута си и вече е активно.
3. **Завършване на маршрут / Актуализиране на статуса на таксито** - Този преход може да върне таксито от "Възложено на маршрут" или "В движение" обратно в "Налично". Това се случва, когато маршрутът е завършен успешно.
4. **Докладване на технически проблем / Маркиране на таксито като недостъпно** - Това събитие може да премести таксито от "Възложено на маршрут" или "В движение" в "В ремонт". Системата отбелязва, че таксито временно не може да изпълнява задачи.
5. **Завършен ремонт / Маркиране на таксито като налично** - Този преход води от "В ремонт" към "Налично". Това показва, че ремонтът е успешно завършен и таксито може да бъде отново използвано.

6. **Невъзможен ремонт / Актуализиране на статуса на таксито** - Този преход прехвърля таксито от "В ремонт" към "Извън услуга". Това означава, че таксито е окончателно изваден от системата.

2.4 Модел на съдържанието/данните

2.4.1 Структура на съдържанието

Системата използва следната структура на съдържанието:

- Потребители: Информация за администратори, началници и шофьори. За всеки потребител се съхраняват:
  - Идентификатор (ID);
  - Име и фамилия;
  - Роля (администратор, началник, шофьор);
  - Данни за контакт (телефон, имейл);
  - Допълнителни атрибути в зависимост от ролята (напр. години опит за шофьори);
- Превозни средства: Данни за автомобилния парк на фирмата, включващи:
  - Регистрационен номер;
  - Марка и модел;
  - Година на производство;
  - Брой места за пътници;
  - Товароподемност;
  - Текущо състояние (на разположение, в ремонт);
  - История на ремонтите.
- Маршрути: Информация за предефинираните маршрути в системата:
  - Уникален идентификатор на маршрута;
  - Точки от маршрута;
  - Разстояние в километри;
  - Разход на гориво;
  - Брой обиколки на ден;
  - Тип превозни средства, подходящи за маршрута.
- Графици: Информация за работните графици на шофьорите, съдържаща:
  - Назначен маршрут;
  - Дата и час на изпълнение;
  - Назначен шофьор;
  - Превозно средство.
- Отчети: Данни за приходите, разходите, ефективността на маршрутите и състоянието на превозните средства:
  - Тип отчет (месечен, годишен);
  - Дата на създаване;
  - Генериран от (администратор или началник);
  - Подробности за разходи, приходи и ефективност.

2.4.2 Връзки между данните

Моделът на съдържанието включва основни връзки:

- Всеки потребител (шофьор) е свързан с множество графици, които включват информация за маршрута и превозното средство.
- Всеки маршрут е свързан с подходящи превозни средства според техните характеристики (напр. товароподемност, брой места).
- Всеки отчет се базира на данни от маршрути, графици и превозни средства.
- Историята на ремонтите за всяко превозно средство се обновява автоматично при промяна на статуса му.

2.4.3 Характеристики и размер на данните

Очакваните обеми и честота на обновяване на данните са:

- Потребители: Около 300 активни потребители, от които 80% са шофьори. Данните за потребителите се обновяват при регистрация, промяна или деактивация.
- Превозни средства: До 100 превозни средства с редовно обновяване на състоянието и историята на ремонтите.
- Маршрути: 200-300 маршрута с възможност за актуализация при промени в инфраструктурата или нуждите на фирмата.
- Графици: Генерират се дневно за всички активни шофьори и се съхраняват за справки.
- Отчети: Създават се автоматично на седмична, месечна и годишна база, като се запазват за исторически анализ.

2.4.4 Формат и кодиране на данните

- Текстови данни: Запазват се в UTF-8 формат за съвместимост и лесно търсене.
- Числови данни: Цели числа за идентификатори, десетични числа за разходи и разстояния.
- Дати и времена: Запазват се в ISO 8601 формат (YYYY-MM-DDTHH:MM:SS).
- Графични данни: Снимки на профили и превозни средства в JPEG или PNG формат.
- Отчети: Генерират се в PDF формат за преглед и споделяне.

3 Дизайн

3.1 Технология за реализация

Реализацията на системата е базирана на набор от технологии, които осигуряват гъвкавост, надеждност и висока производителност. Основните компоненти са:

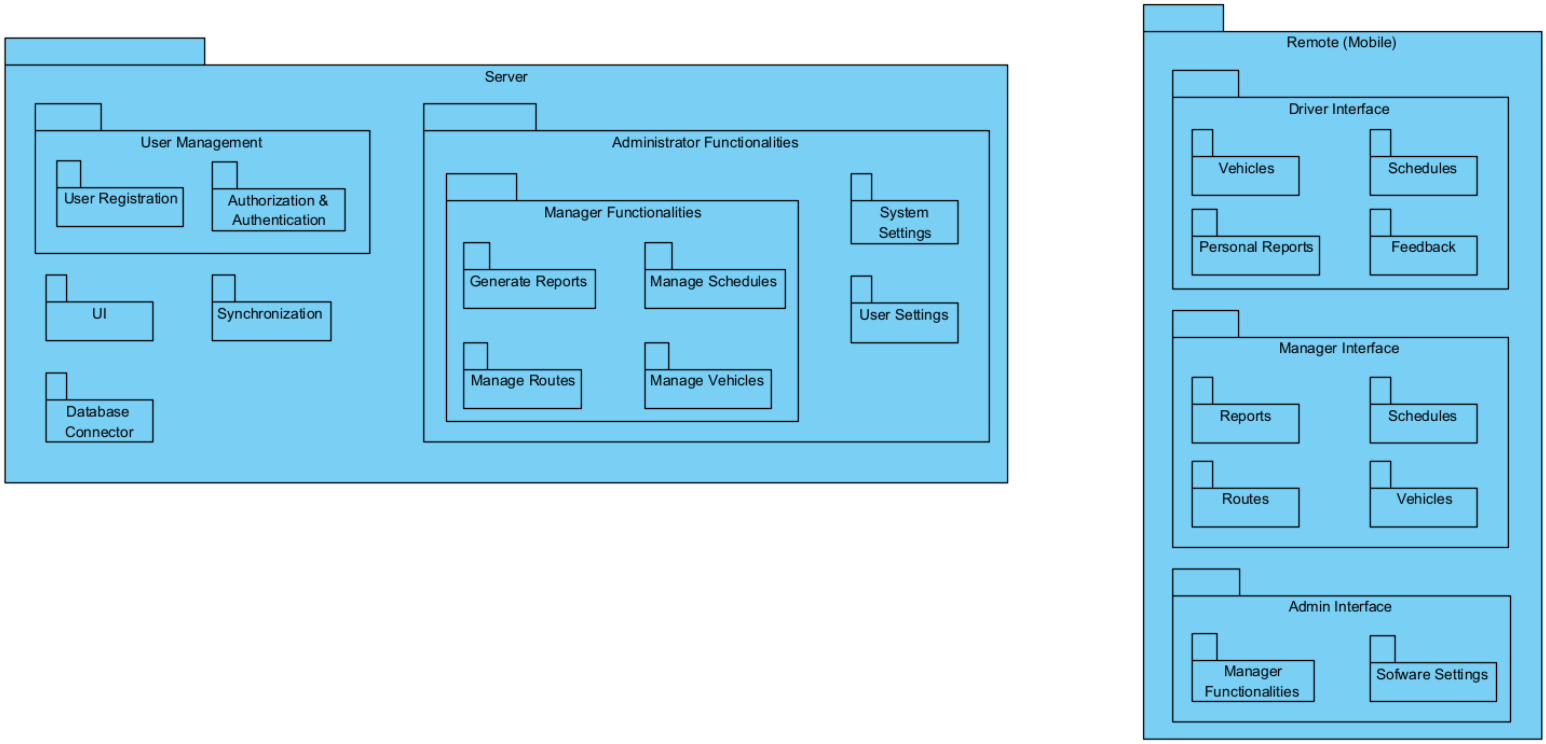
- **Софтуерна платформа**: Системата използва Java като основен език за програмиране. Изборът на Java се основава на нейната стабилност, широка поддръжка и съвместимост с различни операционни системи. Основните рамки и библиотеки включват Spring Boot (за създаване на RESTful API и управление на бизнес логиката), както и Hibernate (за управление на връзките с базата данни и използване на Object-Relational Mapping).
- **База данни**: релационна база данни PostgreSQL. Основните причини за този избор са: поддръжка на сложни заявки и индекси за бързо извличане на данни; възможност за съхранение на големи обеми данни; вградени механизми за сигурност и репликация.
- **Клиентски интерфейс**: Потребителският интерфейс е изграден с помощта на React.js, който предлага динамична и интуитивна работа с приложението. Използваните технологии за клиентската част включват: React Router, Axios, Bootstrap и др.
- **Архитектура**: Системата е проектирана с тристепенна (layered/3-tier) архитектура, която разделя приложението на следните основни слоеве:
    - Представяне: Клиентската част, която обработва взаимодействията с потребителите;
    - Бизнес логика: Сървърната част, която управлява правилата и операциите на системата;
    - Данни: Базата данни, където се съхраняват всички данни на приложението.
- **Среда за разработка**: Разработката се извършва в среда IntelliJ IDEA за сървърната част и Visual Studio Code за клиентската част. Управлението на зависимостите и изграждането на проекта се осъществява с помощта на Maven (за Java) и npm (за React.js).
- **Деплоймънт и хостинг**: Системата ще бъде хоствана в облачна среда AWS, която осигурява висока мащабируемост и надеждност. Основни услуги включват: EC2 (за хостване на сървърната част); RDS (за управление на базата данни); S3 (за съхранение на файлове и отчети).

Тази технология за реализация гарантира, че системата ще бъде устойчива, производителна и лесно мащабируема.

3.2 UML Package Diagram (Модулна декомпозиция)

Модулната декомпозиция на една софтуерна система е основен компонент от цялостната ѝ архитектура. Основната ѝ цел е да се представи т.нар. “разделяне на отговорностите”, което улеснява разбирането на системата от всички заинтересовани страни, подпомагайки значително последващата реализация, употреба и поддръжка.

UML Package диаграмата е сред най-подходящите методи за визуалното ѝ представяне. В този случай, връзки между компонентите/модулите не са показани на схемата, за да не се получи затрудняващо четящия претрупване, а са описани към текстовото описание съпътстващо диаграмата.



Декомпозицията на модулите на системата се състои от два основни компонента - Сървър и отдалечена част (отдалечено устройство). Те комуникират непрестанно, което и обуславя нуждата от интернет свързаност на отдалеченото устройство.

Всички данни, с които работи системата, се съхраняват в базата данни, като връзката с нея се осъществява чрез модула “Database Connector”. Модулът “User Management”осигурява функционалност за регистрация на потребителите (User Registration), вписване в системата, ролите им и правата за достъп (Authorization & Authentication). В модула “UI” е реализирана логиката по потребителския интерфейс към системата. За да бъдат изпълнени нефункционалните изисквания за надеждност на системата, се прави репликация на сървъра и базата данни, като модулът

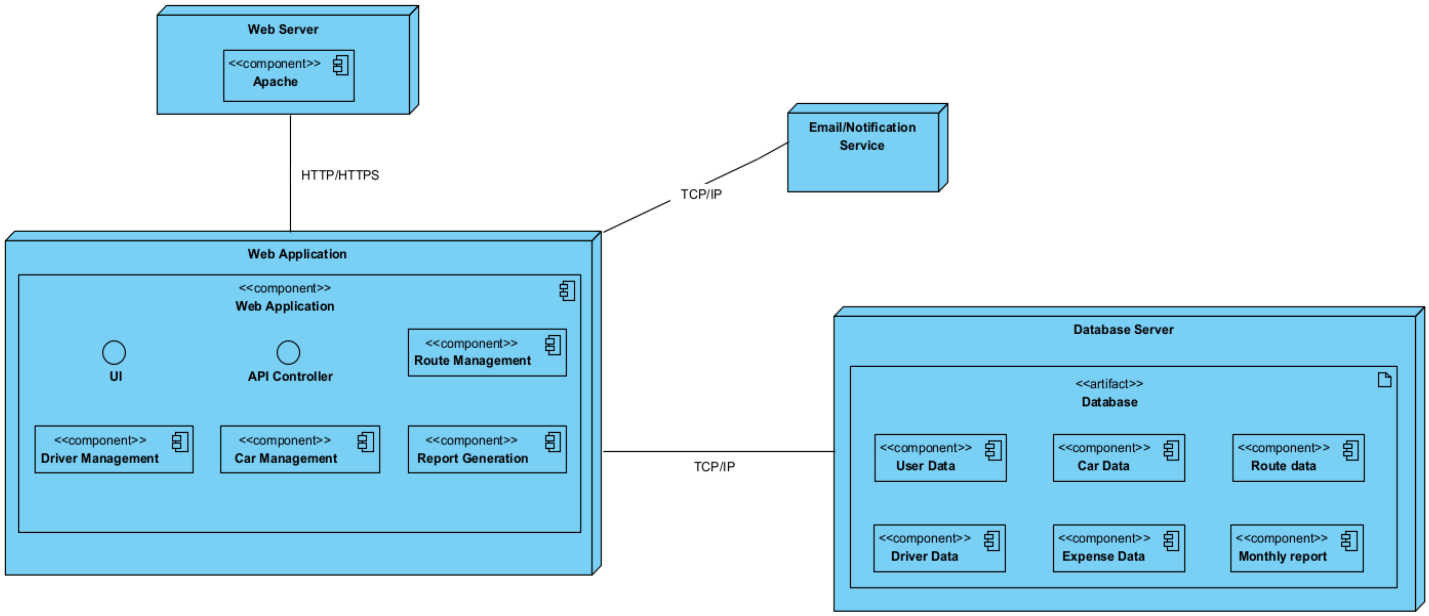
“Synchronization” има задача да осигури консистентност на данните между различните активни копия на системата.

Основната бизнес логика е реализирана в модула “Administrator Functionalities”, тъй като той има достъп до всичко, явявайки се superuser. Ексклузивните модули за него са “System Settings” и “User Settings”, чрез които се управляват всички възможни системни конфигурации и потребителски настройки през отдалечената част, в която пряко свързаните с тези функционалности на администратора са подмодулите на модул “Admin Interface”.

Останалите съставни части на администратора съвпадат с пълната функционалност на ролята “мениджър” - модул “Manager Functionalities”. Там, подмодулите са: “Generate Reports”, “Manage Schedules”, “Manage Routes”, “Manage Vehicles”. В тях се извършва логиката, която е необходима след действия на началниците чрез аналогичните подмодули на “Manager Interface” от “Remote” частта.

Друг основен компонент на мобилния модул е “Driver Interface”. Както името му издава, чрез него шофьорите имат възможност да обновяват информация за своите превозни средства и разписания (графици), като заявките, които могат да подават за промяна на графика си се подават за одобрение от мениджър. Също така могат да извличат лични отчети (Personal Reports) с чисто информативна цел, както и да дават обратна връзка относно маршрут, който са минали.

3.3 UML Deployment Diagram



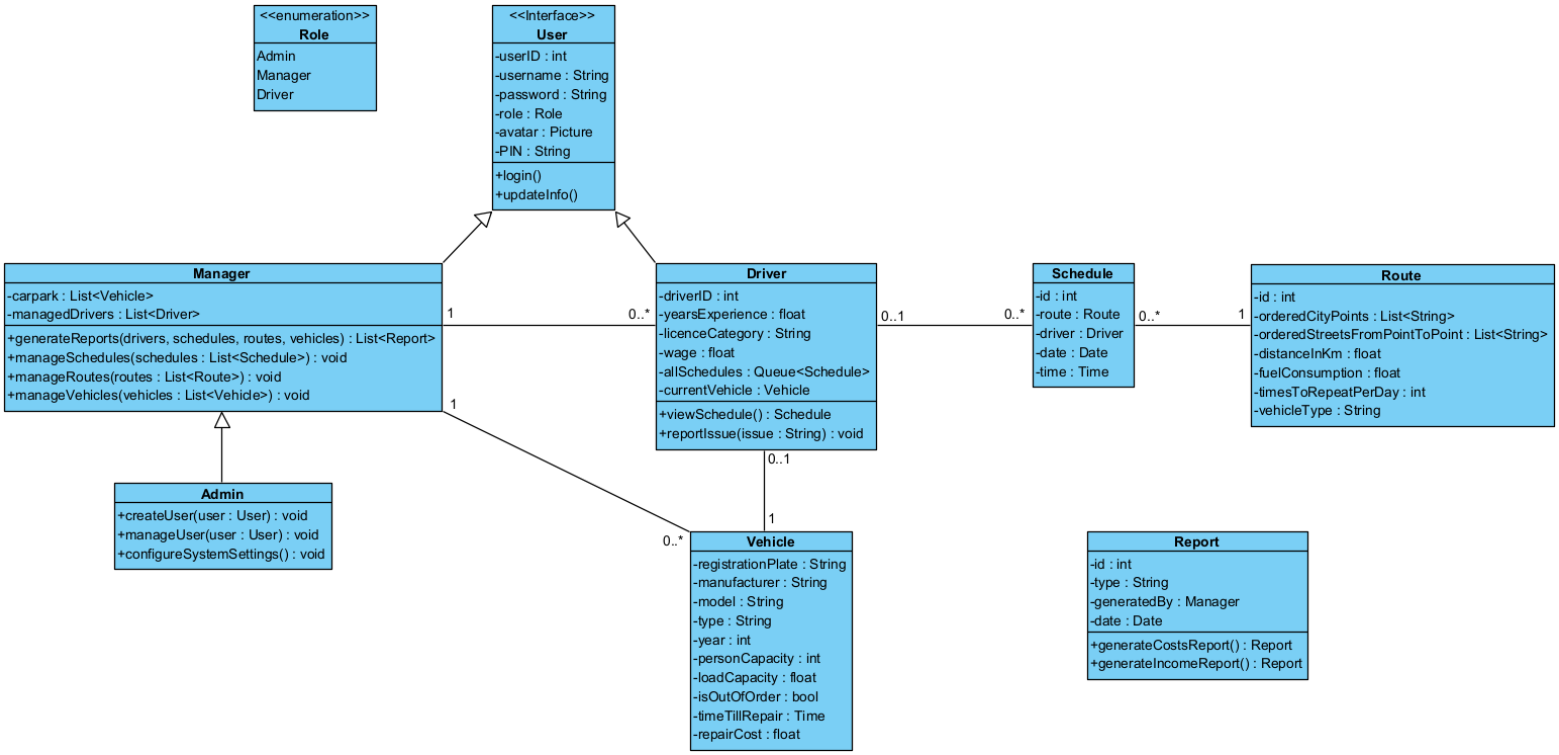
Диаграмата на внедряването (разположението) илюстрира архитектурата на система ни. Тя показва как различните компоненти взаимодействат както помежду си, така и със свързани външни услуги.

- Описание на компонентите
  - Web Server (Apache): служи за управление на HTTP/HTTPS заявки между клиентите и уеб приложението. Това е интерфейсът, който позволява на

потребителите да достъпват системата през браузър или друго клиентско приложение.
    - Web Application: Основният модул за управление на функционалностите. Включва следните компоненти:
        - UI (User Interface): Интерфейс за взаимодействие с потребителите.
        - API Controller: Посредник, който обработва заявките от потребителския интерфейс и ги препраща към съответните модули.
        - Driver Management: Управлява информацията за шофьорите (добавяне, актуализация, и изтриване).
        - Car Management: Управлява данни за автомобилите, като състояние, капцитет и ремонт.
        - Route Management: Управлява маршрутите, като добавяне на възлови точки, дължина и разходи за гориво.
        - Report Generation: Генерира отчети, като приходи, разходи, състояние на автомобилите и ефективност на маршрутите.
    - Database Server: Хранилище за данни, което съхранява всички необходими данни за функционирането на системата:
        - User Data: Данни за потребителите, включително администратори, мениджъри и шофьори.
        - Driver Data: Данни за шофьорите, като име, ЕГН, професионален опит и текущи маршрути.
        - Car Data: Информация за автомобилите, включително текущо състояние и спецификации.
        - Route Data: Данни за маршрутите, като възлови точки, дължина и разходи.
        - Expense Data: Финансови данни за разходите.
        - Monthly Report: Обобщени месечни отчети за разходите и приходите.
    - Email/Notification Service: Компонент, който изпраща уведомления (напр. напомняния за ремонти, графици и отчети) чрез TCP/IP връзка.
- Връзки между компонентите
    - Web Server ↔ Web Application: HTTP/HTTPS протокол за взаимодействие между уеб сървъра и уеб приложението.
    - Web Application ↔ Database Server: TCP/IP връзка за съхранение и извличане на данни.
    - Web Application ↔ Email/Notification Service: TCP/IP връзка за управление на известия.

Тази диаграма илюстрира основната архитектура на системата, като демонстрира ключовите компоненти и начина им на взаимодействие. Тя помага за разбиране на разпределението на отговорностите между отделните части на системата и осигурява ясна основа за разработка, тестване и поддръжка.

3.4 UML Class Diagram



UML диаграмата на класовете представя основните класове, техните атрибути, методи и връзките помежду им в нашата система. Основните компоненти са:

- **Role (Enum):** Изброим тип, който определя ролите на потребителите в системата: Admin, Manager, и Driver.
- **User (Interface):** Представлява базовия интерфейс за всички потребители на системата.
    - **Атрибути:**
        - userID (int): Уникален идентификатор на потребителя.
        - username (String): Име за вход в системата.
        - password (String): Парола за вход.
        - role (Role): Ролята на потребителя (администратор, мениджър или шофьор).
        - avatar (Picture): Снимка на профила.
        - PIN (String): ЕГН.
    - **Методи:**
        - login(): Извършва вход в системата.
        - updateInfo(): Актуализира информацията на потребителя.
- **Manager (Class):** Наследява User и добавя функции за управление на ресурси и създаване на отчети.
    - **Атрибути:**
        - carpark (List<Vehicle>): Списък с превозни средства, управлявани от мениджъра.

        - managedDrivers (List<Driver>): Списък с шофьори, управлявани от мениджъра.
    - **Методи:**
        - generateReports(drivers, schedules, routes, vehicles): List<Report>: Създава отчети.
        - manageSchedules(schedules: List<Schedule>): void: Управлява графици.
        - manageRoutes(routes: List<Route>): void: Управлява маршрути.
        - manageVehicles(vehicles: List<Vehicle>): void: Управлява превозни средства.
- **Admin (Class):** Наследява Manager и добавя допълнителна функционалност, която е специфична за администраторите.
    - **Методи:**
        - createUser(user: User): void: Създава нов потребител в системата.
        - manageUser(user: User): void: Управлява потребителите.
        - configureSystemSettings(): void: Конфигурира настройките на системата.
- **Driver (Class):** Наследява User и добавя специфични за шофьорите атрибути и методи.
    - **Атрибути:**
        - driverID (int): Уникален идентификатор на шофьора.
        - yearsExperience (float): Години опит.
        - licenceCategory (String): Категория на шофьорската книжка (за тип прев.ср).
        - wage (float): Заплата.
        - allSchedules (Queue<Schedule>): Опашка с графици.
        - currentVehicle (Vehicle): Превозното средство, което използва в момента.
    - **Методи:**
        - viewSchedule(): Schedule: Преглежда графика си.
        - reportIssue(issue: String): void: Докладва за проблем.
- **Vehicle (Class)**: Представлява превозно средство в системата.
    - **Атрибути:**
        - registrationPlate (String): Регистрационен номер.
        - manufacturer (String): Производител (Марка).
        - model (String): Модел.
        - type (String): Тип превозно средство.
        - year (int): Година на производство.
        - personCapacity (int): Брой места за пътници.
        - loadCapacity (float): Товароподемност.
        - isOutOfOrder (bool): Статус дали е в неизправност.
        - timeTillRepair (Time): Време до ремонт.
        - repairCost (float): Цена на ремонта.
- **Schedule (Class):** График за шофьорите
    - **Атрибути:**
        - id (int): Уникален идентификатор.
        - route (Route): Маршрутът в графика.
        - driver (Driver): Шофьорът, който е назначен.
        - date (Date): Дата на графика.
        - time (Time): Време на изпълнение.

- **Route (Class):** Маршрут за превозни графиците на шофьорите.
    - **Атрибути:**
        - id (int): Уникален идентификатор.
        - orderedCityPoints (List<String>): Поредица от възловите точки в града (върхове в графа).
        - orderedStreetsFromPointToPoint (List<String>): Улиците между тези точки (ребра в графа).
        - distanceInKm (float): Разстояние на пътя в километри.
        - fuelConsumption (float): Среден разход на гориво за изминаване.
        - timesToRepeatPerDay (int): Колко пъти маршрутът трябва да се изпълнява дневно.
        - vehicleType (String): Типът превозно средство, което може да извърши маршрута.
- **Report (Class):** Отчет, генериран от мениджър или администратор.
    - **Атрибути:**
        - id (int): Уникален идентификатор.
        - type (String): Тип на отчета.
        - generatedBy (Manager): Само Manager, в частност Admin, генерира отчет.
        - date (Date): Дата на създаване.
    - **Методи:**
        - generateCostsReport(): Report: Генерира отчет за разходите.
        - generateIncomeReport(): Report: Генерира отчет за приходите.

Връзките между класовете са следните:

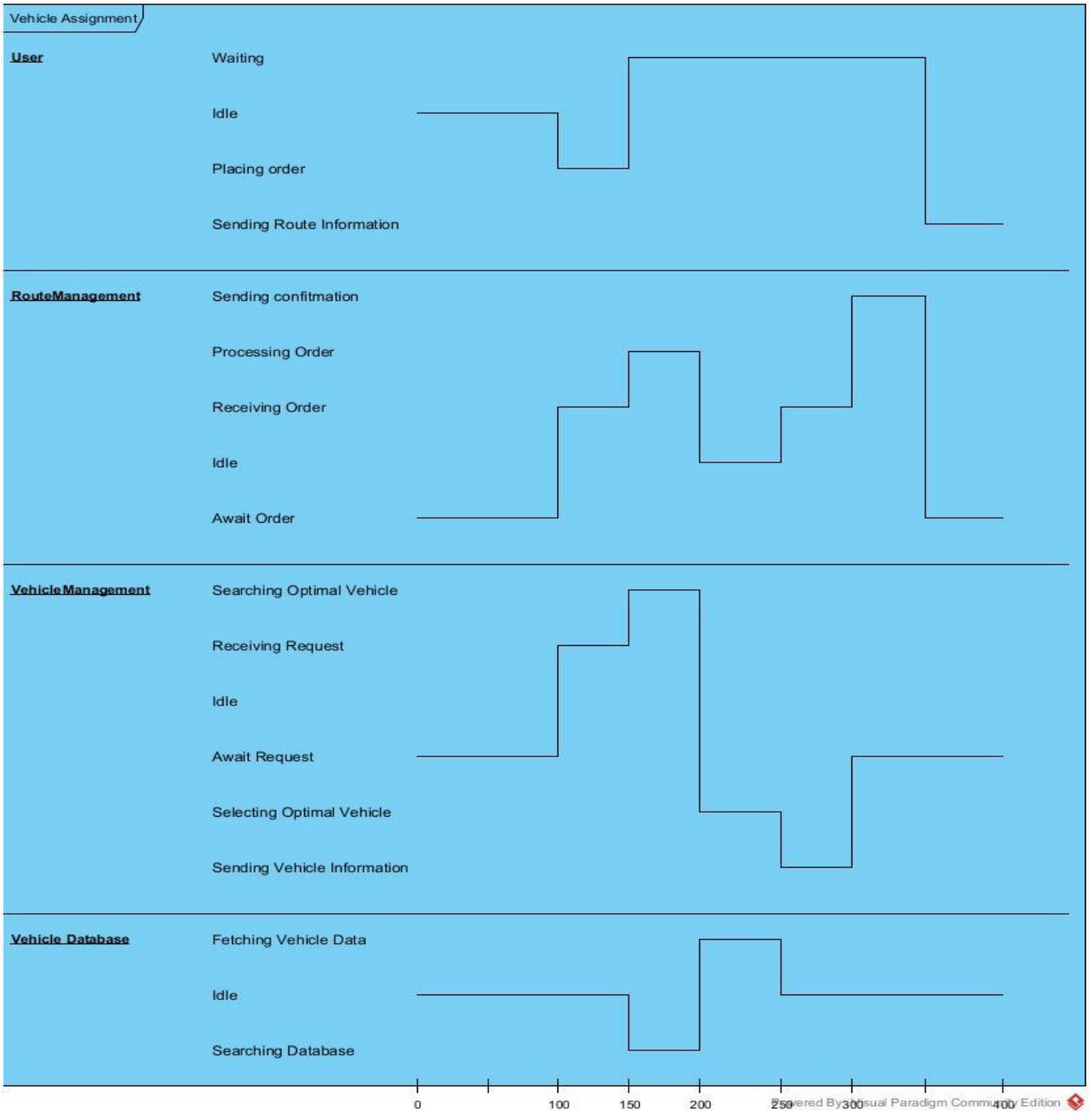
- Manager (1) — (0..*) Driver
- Manager (1) — (0..*) Vehicle
- Schedule (0..*) — (1) Route
- Schedule (0..*) — (0..1) Driver.
- Driver (0..1) — (1) Vehicle

<u>Ограничения, описани на Object Constraint Language (OCL)</u>

- **context** Driver **inv**: self.currentVehicle->size() = 1 // шофьор винаги трябва да разполага с 1 кола
- **context** Driver **inv**: self.licenceCategory = self.currentVehicle.type // категорията на шофьора трябва да отговаря на тази на назначеното му превозно средство
- **context** Schedule **inv**: self.route.vehicleType = self.driver.currentVehicle.type // за всяко разписание трябва превозното средство нужно за маршрута му да отговаря на това на назначения му шофьор
- **context** Schedule **inv**: not self.route.oclIsUndefined() // всяко разписание трябва да има дефиниран маршрут
- **context** Manager **inv**: self.managedDrivers->notEmpty() and self.carpark->notEmpty() // мениджър трябва да разполага с множество коли и с множество подчинени шофьори

- **context** Vehicle **inv**: self.registrationPlate.size() > 0 // всяко прев. средство трябва да има регистрационен номер
- **context** Schedule **inv**: not self.date.oclIsUndefined() and not self.time.oclIsUndefined() // не може да се създаде график без дата и час
- **context** Driver **inv**: self.yearsExperience >= 5 // наемат се шофьори с поне 5 години опит
- **context** Route **inv**: self.distanceInKm > 0.5 // не се правят курсове за под половин км

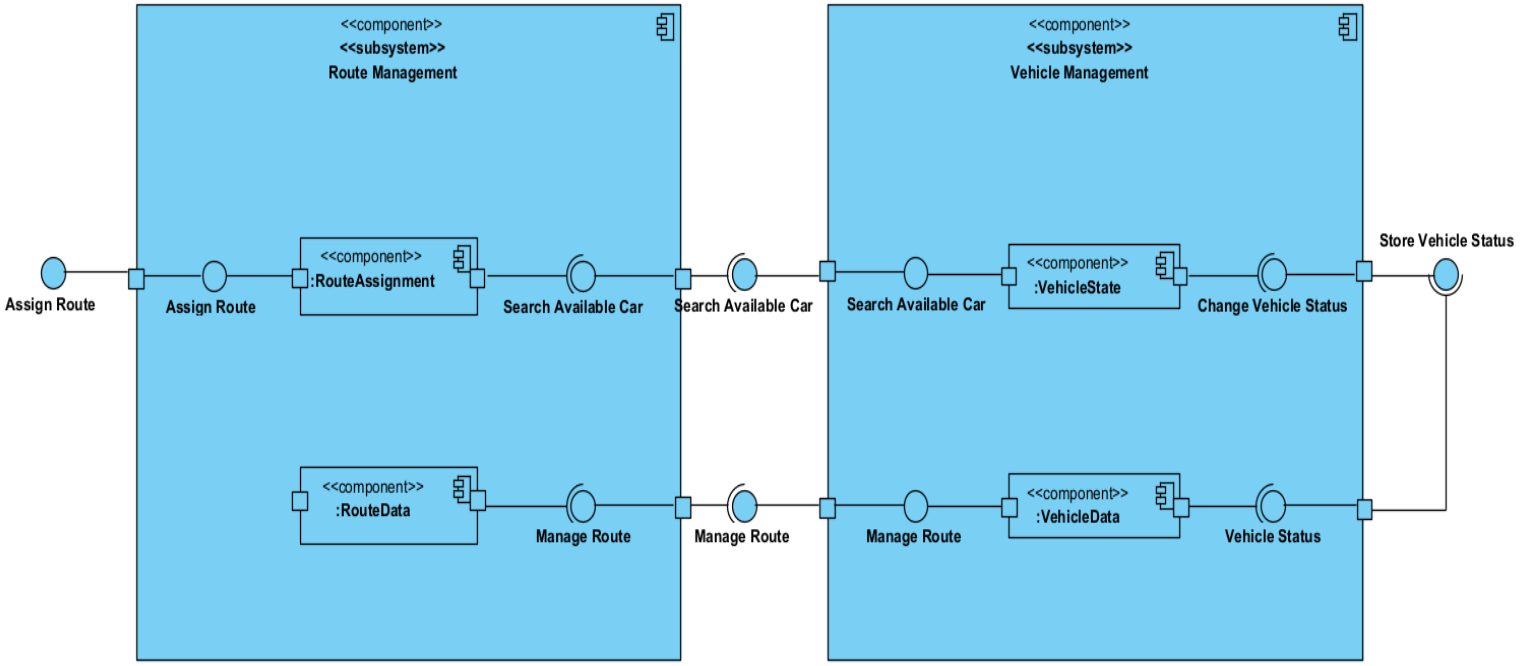
3.5 UML Timing Diagram



Тази диаграма представлява времева диаграма на система за назначаване на превозни средства. Диаграмата показва състоянията на всеки компонент и преходите между тях във времето, измерено в милисекунди. Имаме следните компоненти:

1. **Потребител**:
    - Изчакване (**Waiting**): В началото на процеса потребителят е в състояние на изчакване, преди да извърши действие.
    - Бездействие (**Idle**): След изчакване потребителят временно не извършва активност.
    - Подаване на заявка (**Placing Order**): Следващият преход е активната стъпка, в която потребителят подава заявка за услуга.
    - Изпращане на информация за маршрута (Sending Route Information): След подаването на заявката потребителят предоставя информация за маршрута.
2. **Управление на маршрути (Route Management)**:
    - Изпращане на потвърждение (**Sending Confirmation**): След като системата получи информация от потребителя, тя изпраща потвърждение.
    - Обработка на заявка (**Processing Order**): Системата обработва заявката, за да идентифицира нуждите на потребителя.
    - Получаване на заявка (**Receiving Order**): След обработка системата записва данните за заявката.
    - Бездействие (**Idle**): Кратък период, в който компонентът е в изчакване.
    - Изчакване на заявка (**Await Order**): Компонентът остава в изчакване, докато се появи нова заявка.
3. **Управление на превозни средства (Vehicle Management):**
    - Търсене на оптимално превозно средство (**Searching Optimal Vehicle**): Този компонент започва процеса на анализ за намиране на най-подходящо превозно средство.
    - Получаване на заявка (**Receiving Request**): След като бъде подадена заявката, системата я получава за обработка.
    - Бездействие (**Idle**): Компонентът преминава в неактивно състояние за кратък период.
    - Изчакване на заявка (**Await Request**): Компонентът чака нова заявка за обработка.
    - Избор на оптимално превозно средство (**Selecting Optimal Vehicle**): След анализ на заявките и информацията се избира най-подходящото превозно средство.
    - Изпращане на информация за превозното средство (Sending Vehicle Information): Накрая информацията за избраното превозно средство се изпраща към останалите компоненти.
4. **База данни за превозни средства (Vehicle Database):**
    - Извличане на данни за превозно средство (**Fetching Vehicle Data**): Започва се с търсене и извличане на необходимите данни от базата.
    - Бездействие (**Idle**): След като данните са извлечени, компонентът преминава в състояние на бездействие.
    - Търсене в базата данни (**Searching Database**): При нова заявка, базата се активира и започва търсене в хранилището

3.6 UML Component Diagram

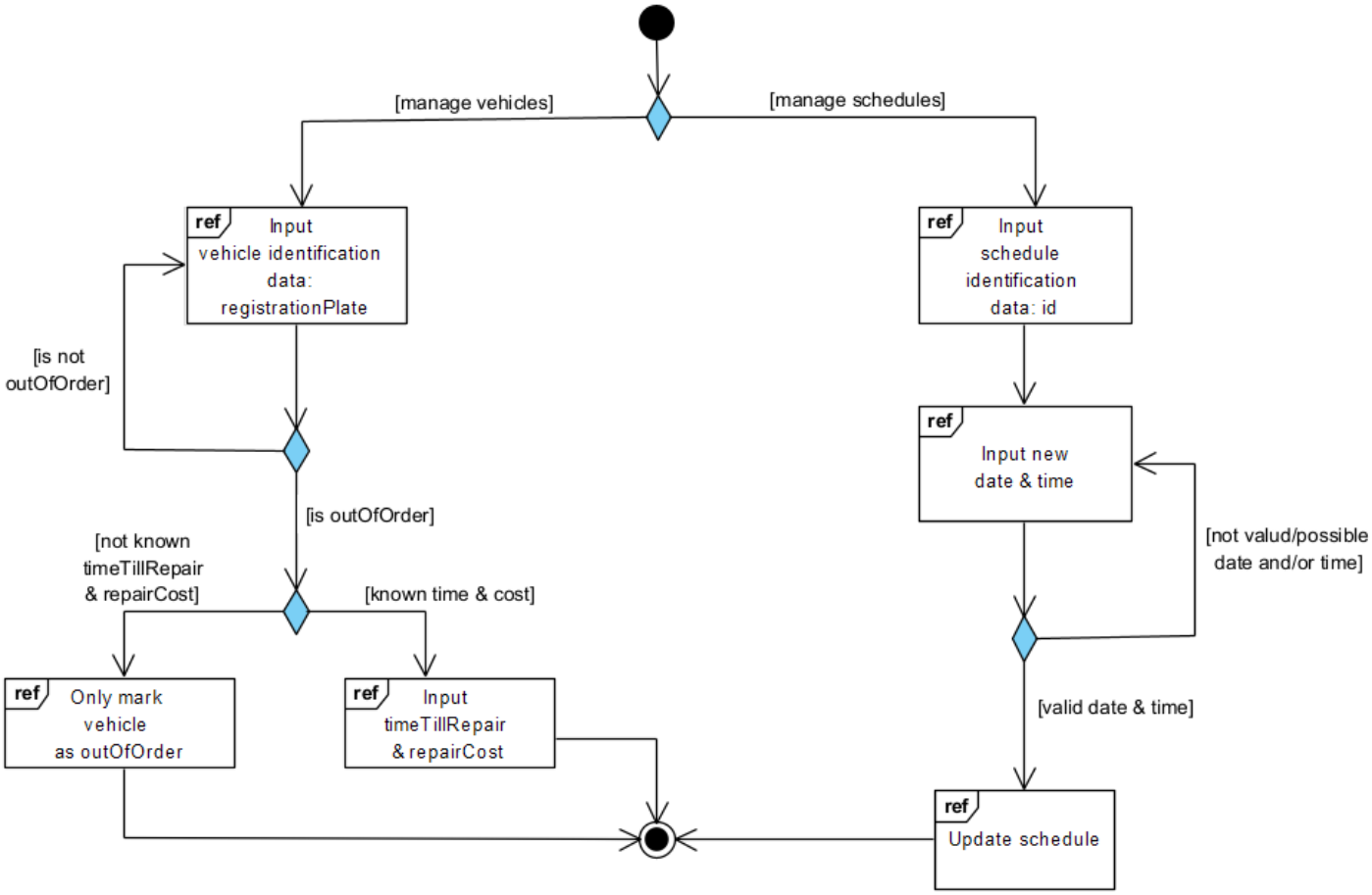


Диаграмата на компонентите визуализира два основни подсистеми: **Управление на маршрути** и **Управление на превозни средства**. Подсистемата "Управление на маршрути" съдържа компонентите *RouteAssignment* и *RouteData*, които отговарят за задаване и управление на маршрути, както и за търсене на налични превозни средства. Подсистемата "Управление на превозни средства" включва компонентите *VehicleState* и *VehicleData*, които следят и актуализират състоянието на превозните средства.

Потокът на данни е организиран, като се използват специфични входове и изходи за:

- Задаване на маршрути (**Assign Route**)
- Управление на маршрути (**Manage Route**)
- Актуализиране състоянието на превозните средства (**Change Vehicle Status**)
- Съхраняване на данни за състоянието на превозните средства (**Store Vehicle Status**).

3.7 UML Interaction Overview Diagram



UML Interaction Overview диаграмата е вид UML диаграма, която комбинира елементи от диаграмите на активност и взаимодействие, за да визуализира последователността на действията и начина, по който те взаимодействат в сложни процеси. Тя се използва за представяне на потоци от събития, които включват както логически решения, така и препратки към по-детайлни взаимодействия между системни компоненти.

Тук диаграмата е приложена, за да се моделира управлението на превозни средства и разписания, от гледна точка на шофьорите.

Пояснение:

1. **Начало на процеса**: Процесът започва с начално състояние (начална точка), което показва началото на изпълнението на възможностите на шофьора в приложението. Опциите му се свеждат до две възможности:
2. **Управление на превозни средства**:
    a. Потребителят трябва да въведе идентификационни данни на превозното средство (регистрационния му номер);
    b. След въвеждането на данните трябва да въведе състоянието на превозното средство:
        i. Ако превозното средство **не е повредено**, връща потребителя на стъпка “a”.

ii. Ако превозното средство **е повредено**, се прави допълнителна проверка дали времето за ремонт и разходите за него са известни. Като ако не са - превозното средство се маркира само като "неизправно", докато ако са - потребителят въвежда тези данни, след което данните в системата са обновени и процесът приключва.

3. **Управление на разписания**:
   a. Потребителят трябва да въведе идентификатор на разписанието (*id*);
   b. След това се въвеждат нови дата и час за разписанието;
   c. Валидността на въведената дата и час се проверява:
      i. Ако **не са валидни или възможни**, шофьорът трябва отново да въведе нова дата и час (стъпка “b”).
      ii. Ако **са валидни**, разписанието се актуализира и процесът приключва.
4. **Край на процеса**: Процесът приключва, когато се изпълнят съответните действия. Обозначено с черна точка с кръг около нея в схемата.

4 Тестване

Тестването на системата е ключов етап от процеса на разработка, който включва различни видове тестове и множество тестови случаи, насочени към осигуряване на правилната работа, производителността и надеждността на приложението.

На първо място, функционалното тестване гарантира, че всички функционалности, описани в спецификациите, работят коректно. Това включва проверки за успешна регистрация на потребители от различни роли, като администратори, началници и шофьори; валидиране на входни данни и изпълнение на операции, като добавяне и управление на графици.

Интеграционното тестване проверява правилната комуникация между клиентската и сървърната част на системата, включително връзките с базата данни и извличането на данни.

Тестването за производителност се фокусира върху измерване на времето за отговор при различни натоварвания, както и способността на системата да обработва големи обеми данни, например до 1000 потребители и 500 маршрута.

Освен всичко това, съвместимоста на приложението е важна, затова се извършват тестове в различни браузъри като Chrome, Firefox и Edge, както и на различни устройства – настолни компютри, таблети и смартфони.

Сигурността също е основен приоритет, като се проверява криптирането на чувствителни данни, като пароли, и се тестват механизми за предотвратяване на неоторизиран достъп.

Тестването включва и генериране на отчети в PDF формат, които се проверяват за коректност и консистентност, както и работа със съхранени данни за история на ремонтите.

Всички тези тестове се извършват в контролирана среда с поддръжка на Windows, Linux и macOS, за да се гарантира, че системата отговаря на всички функционални и нефункционални изисквания.

5 Заключение и възможно бъдещо развитие

Разработената система успешно постига основната си цел – автоматизация и оптимизация на процесите, свързани с управлението на таксиметрови услуги. Чрез създаването на модулна архитектура и внедряването на основни функционалности като управление на потребители, маршрути, автомобили и генериране на отчети, системата осигурява ефективност, гъвкавост и удобство за различните типове потребители. Въпреки срещнатите предизвикателства по време на разработката, като сложността на интеграцията между компонентите и осигуряването на надеждна обработка на данни, проектът успешно демонстрира устойчивост и добро проектиране. Системата създава стабилна основа за управление, като едновременно с това предлага мащабируемост за бъдещи разширения.

Системата има голям потенциал за разширяване и подобрения, които биха могли допълнително да увеличат нейната стойност и ефективност. Например, могат да се добавят мобилни приложения за шофьори и клиенти, интеграция с GPS и навигационни системи за оптимизация на маршрути в реално време, както и разширени аналитични функции за прогнозиране на разходи и ефективност. С оглед на развитието, ако в бъдеще функционалностите на “Admin” и “Manager” станат твърде различни, би било подходящо част от функциите на Manager да бъдат отделени в специфичен интерфейс (например Manageable), който само той да имплементира. Освен това, сигурността и потребителският интерфейс могат да бъдат значително подобрени чрез по-модерни методи за криптиране и адаптивен дизайн за различни устройства. Тези стъпки ще позволят на системата да се адаптира към променящите се нужди на потребителите и на пазара, осигурявайки дългосрочна устойчивост и ефективност.

6 Разпределение на работата

За краткост ползваме следните съкращения: Радослав - Р; Кристиан - К.

1 Въведение	Р
2 Анализ на решението	Р+К
2.1 Потребителски изисквания и работен процес	Р+К
2.1.1 Функционални изисквания	Р
2.1.2 Нефункционални изисквания	К
2.2 Примерен графичен интерфейс	К
2.3 Диаграми на анализа	Р+К
2.3.1 Entity-Control-Boundary Pattern	Р
2.3.2 UML Use Case Diagram	К
2.3.3 UML Activity Diagram	К

2.3.4 UML Sequence Diagram	Р
2.3.5 UML Communication Diagram	К
2.3.6 UML State Machine Diagram	К
2.4 Модел на съдържанието/данните	Р+К
2.4.1 Структура на съдържанието	Р
2.4.2 Връзки между данните	Р
2.4.3 Характеристики и размер на данните	К
2.4.4 Формат и кодиране на данните	К
3 Дизайн	Р+К
3.1 Технология за реализация	Р
3.2 UML Package Diagram (Модулна декомпозиция)	Р
3.3 UML Deployment Diagram	К
3.4 UML Class Diagram	Р
3.5 UML Timing Diagram	К
3.6 UML Component Diagram	К
4 Тестване	Р
5 Заключение и възможно бъдещо развитие	Р+К

7 Използвани литературни източници и Уеб сайтове

1. Всички лекциите качени в Мудъл курс: Обектно-ориентиран анализ и проектиране на софтуерни системи, зимен семестър 2024/2025 на адрес: https://learn.fmi.uni-sofia.bg/course/view.php?id=10789;
2. https://en.wikipedia.org/wiki/Entity-control-boundary;
3. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-communication-diagram/;
4. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/;
5. https://en.wikipedia.org/wiki/Spring_Boot;
6. https://en.wikipedia.org/wiki/PostgreSQL;
7. https://en.wikipedia.org/wiki/React_(software);
8. https://bg.wikipedia.org/wiki/ISO_8601;

9. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-timing-diagram/;
10. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/;
11. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-interaction-overview-diagram/.