**“LogiTrack”**

**Софтуер за логистични решения**

**(архитектурен проект)**

**Проект номер:** 18  
**Автори:** Йоана Йотова, Калин Герджиков, Илия Кръстев, Денис Патронов  
**Дата:** 05.11.2024г.

**Въведение**

Този документ представя архитектурата на проекта „LogiTrack“ – софтуерно решение за оптимизация на логистичните процеси в малки и средни фирми. Проектът цели подобрение на комуникацията между логистичните компании и техните клиенти, управление на заявки и оферти, проследяване на поръчки, както и обработка на данни за транспортни средства и разходи.

Софтуерът осигурява централизирана система за управление на процесите – от регистрация на клиенти и заявки до визуализация на финансови данни и генериране на отчети. Компаниите ще могат ефективно да управляват своите логистични дейности и да предоставят на клиентите удобни опции за проследяване и статистики.

Екипът е разпределен както следва:

* **Бекенд разработка**: Йоана Йотова – бизнес логика, защита, бази данни (ASP.NET).
* **Фронтенд разработка**: Денис – потребителски интерфейс (HTML, CSS, JavaScript, Razor).
* **Тестване**: Илия – юнит тестване, ръчно тестване, оценка на потребителското изживяване.
* **Документация и визуализация**: Калин – UML диаграми, документация, идеи за интерфейса, организация на структури от бази данни, инициализация.

Чрез анализ на нуждите на логистични фирми и клиенти, екипът формира функционалности, които отговарят на практическите изисквания. Проектът се координира чрез **Discord** за обсъждания и **GitHub Projects** за управление на задачите. Използваната методология осигурява гъвкавост, позволявайки паралелна работа по бекенд, фронтенд и тестване, като накрая се интегрират и финално проверяват всички компоненти.

Проектът „LogiTrack“ цели да улесни работата на логистичните компании, предоставяйки модерен и ефективен инструмент, който максимално отговаря на нуждите на клиентите.

**Предназначение**

Проектът е създаден с цел да отговори на нуждите на малки и средни логистични компании, които искат да подобрят ефективността на своите операции и да автоматизират различни аспекти на логистичния процес. Софтуерното решение „LogiTrack“ предоставя удобна и централизирана платформа, която обхваща всички ключови стъпки – от регистрацията на клиенти и заявки до управлението на доставки и финансови отчети.

**Основни цели:**

* **Автоматизация на работните процеси**:  
  Системата автоматизира редица стандартни операции, като генериране на отчети, изчисляване на цени за оферти и проследяване на доставките. Това спестява време и ресурси на логистичните фирми, които могат да се фокусират върху стратегическите задачи и подобряване на услугите си.
* **Улеснено проследяване на доставките**:  
  Клиентите получават възможност в реално време да следят статуса на своите доставки и местоположението на транспортните средства. Това елиминира необходимостта от постоянна комуникация с персонала и подобрява прозрачността на целия процес.
* **Подобрена комуникация с клиентите**:  
  Създава се централизирана и организирана система за комуникация, която улеснява фирмите при предоставяне на оферти и информиране на клиентите за статуса на техните заявки. Това подобрява цялостното обслужване и удовлетворението на клиентите.
* **Ефективно управление на ресурси**:  
  Чрез регистриране на транспортни средства и профили на служителите, системата позволява лесен мониторинг на наличните ресурси. Така се предотвратяват проблеми като забавяния поради липса на поддръжка на камиони или изтекли лицензи.
* **Прозрачност и отчетност**:  
  Системата предлага възможност за генериране на финансови статистики и отчети, което позволява на компаниите да следят ефективността на своите операции. Това улеснява управлението на разходите и планирането на бъдещи дейности.
* **Подкрепа за управленски решения**:  
  Платформата предоставя детайлни аналитични данни и статистики за заявките, приходите и разходите, което помага на мениджърите да вземат по-добри и информирани решения. Това е особено полезно в динамични ситуации, където бързата реакция е от съществено значение.
* **Подобрена организационна ефективност**:  
  Събирането на различни функции в една платформа спомага за по-добра координация между отделите в една фирма – от логистичния персонал до счетоводителите и шофьорите. Така се постига по-бързо изпълнение на задачите и по-добра организация на процесите.
* **Гъвкавост и адаптивност**:  
  Софтуерът е проектиран така, че да може лесно да бъде разширяван с нови функции, ако фирмата се разраства или се появят нови бизнес нужди. Тази гъвкавост гарантира, че системата ще остане полезна и актуална в дългосрочен план.
* **Интеграция с външни услуги**:  
  Системата е интегрирана с популярни платформи като Google Maps за проследяване на маршрутите, Google Drive за съхранение на документи и Twilio за изпращане на уведомления по имейл. Това улеснява работата както за служителите, така и за клиентите, осигурявайки бърз и лесен достъп до необходимата информация.
* **Сигурност на данните**:  
  Високото ниво на защита на данните е ключов аспект на системата. Използват се криптирани канали за комуникация и стриктни процедури за контрол на достъпа, което гарантира, че информацията на фирмите и клиентите остава конфиденциална.
* **Подобрено клиентско обслужване**:  
  Клиентите могат да следят своите заявки, да получават отчети и да имат достъп до детайлни статистики. Това не само подобрява тяхното изживяване, но и изгражда доверие и дългосрочни отношения между клиентите и логистичната фирма.

Софтуерът „LogiTrack“ предоставя на логистичните компании удобен, сигурен и ефективен инструмент, който подобрява тяхната работа и удовлетворението на клиентите. Той е ключ към по-добра организация и конкурентоспособност в динамичната сфера на логистиката.

**Проблеми, които решаваме**

* Разпокъсан процес и липса на централизирана информация: Традиционните методи за управление на логистични процеси често разчитат на ръчно въвеждане на информация и дори използване на множество различни платформи, което води до забавяния и грешки. Нашата система предлага единна платформа, която обединява всички нужни функции и данни.
* Прозрачност за клиентите: Възможността за клиентите да проследяват своите поръчки и да имат достъп до информация за техните заявки значително подобрява доверието и удовлетворението им.
* Ограничена ефективност и контрол на ресурсите: Със системата се въвежда ефективен начин за управление на ресурсите, което осигурява информация за състоянието на транспортните средства и служителите, избягва забавяния и подобрява ефективността.

Този проект не само модернизира процесите на логистичните компании, но и изгражда дългосрочна стойност чрез по-добро обслужване и подобрено управление на ресурсите.

**Обхват**

Документът обхваща всички основни етапи от разработката на проекта – от началната идея до поддръжката на готовия продукт. Проектът започва с консултации с логистични фирми, за да се изяснят техните нужди, проблеми и желани функционалности, които да подобрят процесите им.

След събиране на информация екипът избира подходящи технологии – ASP.NET за бекенда и HTML, CSS и JavaScript за фронтенда. Тези технологии осигуряват стабилна и лесна за използване система. В допълнение, услугите на Google Maps и Twilio се интегрират за улесняване на проследяването и уведомленията.

Работата по проекта се разделя между членовете на екипа според техните умения – бекенд, фронтенд, тестване и документация. Това позволява организирана и ефективна разработка.

**Основни стъпки:**

1. **Разработка и тестване**: Паралелно с разработката се провежда ръчно тестване на всички модули, за да се гарантира правилната им работа в реални условия.
2. **Интерфейс**: Потребителският интерфейс е проектиран да бъде интуитивен и лесен за навигация, за да осигури добро изживяване на потребителите.
3. **Документация**: Описват се всички функционалности и процеси, за да се улесни бъдещата поддръжка.
4. **Поддръжка**: Разработва се план за редовни актуализации и обучения на служителите за ефективна работа със софтуера.

**Заключение:**

Проектът обхваща всички важни етапи от създаването на идеята до поддръжката, с цел предоставяне на функционална, удобна и ефективна система за логистичните компании.

**Актьори**

Проектът включва няколко ключови групи участници, които използват софтуера, за да улеснят работата си:

1. **Клиенти (фирми)** – Те подават заявки за транспорт, изискват оферти и проследяват доставки. Чрез приложението лесно намират информация за статуса и местоположението на товарите си, което осигурява прозрачност и улеснение.
2. **Логистични оператори (спедитори)** – Управляват маршрути, разпределят товари и регистрират шофьори и камиони. Системата им помага да следят товарите в реално време, да създават оферти и да оптимизират разходите.
3. **Шофьори** – Актуализират статуса на доставките (натоварени, разтоварени, доставени). Те също така имат достъп до информация за предстоящите си и минали доставки.
4. **Счетоводители** – Използват софтуера за отчети за разходи и приходи, управление на документи и анализ на печалбите и загубите по маршрути.
5. **Административен персонал** – Следят отчети, данни за камиони, ефективността на шофьорите и финансовите резултати. Това помага за стратегически решения и подобряване на логистичните операции.

Софтуерът улеснява работата и координацията между всички участници, като води до по-ефективни услуги, оптимизация на ресурсите и удовлетворени клиенти.

**Използвани термини и символи**

В рамките на проекта се използват ключови термини и съкращения, които са важни за разбирането на софтуера и неговите функции:

1. **Геолокация** – Технология за определяне на местоположението чрез GPS координати, използвана за проследяване на товарите в реално време.
2. **Google Cloud** – Услуга за облачно съхранение и управление на данни, включително интеграция с Google Maps API.
3. **Палет** – Стандартна платформа за съхранение и транспорт на стоки (индустриален и евро размер).
4. **Коефициент на цена** – Параметър за изчисляване на цената на транспортна услуга, базиран на разстояние и други условия.
5. **Спедитор** – Лице или компания, която организира транспорта на стоки, като координира между клиенти и транспортни компании.
6. **Инвойс** – Документ, който съдържа информация за продажбата на стоки или услуги, цената им и условията за плащане.
7. **Транспортна заявка** – Документ, който съдържа информация за поръчаната транспортна услуга – товар, маршрут, спецификации.
8. **Доставка** – Процесът на преместване на стоки от едно място до друго в рамките на уговорени срокове и условия.
9. **Клиентски профил** – Информация за клиент, включваща история на поръчките и специални изисквания.
10. **Анализ на разходите** – Оценка на транспортните разходи с цел тяхната оптимизация и подобряване на ефективността.
11. **Фронтенд** – Част от софтуера, която потребителите виждат и използват; интерфейсът на приложението.
12. **Бекенд** – Част от софтуера, която обработва данните и логиката; включва сървъри, бази данни и вътрешни процеси.

Тези термини са от ключово значение за разбирането на системата и нейната работа.

**Нефункционални изисквания**

1. **Достъпност**
   * **Откриване на проблеми**: Ще използваме инструменти като *Grafana* и *Prometheus* за наблюдение на сървърите, натоварването на базата данни MSSQL и критични събития. Ако хостваме в Azure, *Azure Monitor* ще осигури базов мониторинг и известия.
   * **Възстановяване**: Ще внедрим автоматизирани резервни копия на MSSQL, които ще позволяват бързо възстановяване в случай на инцидент.
2. **Разширяемост**
   * **Модулярност и абстракция**: Основните функции на системата ще бъдат разделени на независими сървиси (например за управление на доставки или плащания). Това ще улесни добавянето на нови функции.
   * **Разделяне на контролери и сървиси**: В рамките на MVC архитектурата всеки контролер ще използва отделни сървиси за бизнес логика, което ще позволи лесна адаптация на кода.
3. **Производителност**
   * **Оптимизация на базата данни**: Ще използваме индексиране на таблиците и инструменти като *MSSQL Profiler* и *Database Engine Tuning Advisor*, за да ускорим заявките и да подобрим времето за реакция на системата.
4. **Сигурност**
   * **Удостоверяване**: Системата ще използва *Identity Framework* за управление на потребителските акаунти и безопасен достъп.
   * **Криптиране**: Данните ще се криптират по време на съхранение и предаване, като паролите ще се хешират.
5. **Възможност за тестване**
   * **Модулни тестове (Unit Tests)**: Използване на *NUnit* за тестване на отделни функции и модули.
   * **Интеграционни тестове**: Приложение на *Selenium* за тестване на потребителския интерфейс от край до край.
6. **Интероперабилност**
   * **Интеграция с външни услуги**: Платформата ще поддържа интеграция с Google API (за маршрути) и Twilio (за известия), осигурявайки допълнителни функционалности.
7. **Използваемост**
   * **Интуитивен интерфейс**: Дизайнът ще бъде лесен за използване, с ясни точки за навигация и визуализация, за да улесни потребителите.
8. **Съвместимост**
   * Системата ще бъде съвместима с всички популярни браузъри.

Тези изисквания гарантират надеждност, гъвкавост и ефективност на софтуера, като покриват всички основни аспекти на функционирането му.

**Архитектурен обзор**

Архитектурата на софтуера е проектирана чрез различни изгледи, които обхващат всички основни аспекти на системата – от функционалностите и структурата до процесите, данните и внедряването. Всеки изглед е описан чрез UML диаграми и текстови обяснения, за да предостави ясна представа за техническите и потребителските изисквания на системата.

**1. Use-case изглед**

Този изглед показва как софтуерът ще бъде използван от различните потребители (клиенти, администратори, счетоводители, спедитори, шофьори) и какви са техните действия в системата. Чрез *Use-case диаграми* се илюстрират основните функционалности, като подаване на заявки, управление на доставки, финансови операции и автоматични процеси.

**2. Логически изглед**

Логическият изглед описва структурата на системата чрез *Class диаграми*, които показват основните класове, техните методи и връзки. Този изглед дава представа за компонентите на архитектурата и как те взаимодействат помежду си, за да изпълняват основните функции.

**3. Процесен изглед**

Този изглед се фокусира върху потока от действия в системата.

* *Activity диаграми*: Показват стъпките в основни операции, като регистрация на клиент, логване, добавяне на статус на доставка и обработка на касови бележки.
* *Communication диаграми*: Представят взаимодействията, например автоматично изпращане на имейли чрез Twilio, качване на документи в Google Drive и изчисляване на разстояния чрез Google Maps API.

**4. Изглед на данните**

Изгледът на данните представя как се съхраняват и обработват данните в системата. Чрез диаграми на база данни се илюстрират класовете, основните полета и връзките между тях, което дава ясна представа за структурата на съхранението и обработката на информацията.

**5. Изглед на внедряването**

*Deployment диаграмите* показват хардуерната инфраструктура на системата, включително сървъри за базата данни, уеб сървър и API. Включени са и потребителските работни станции, както и комуникационните канали (интернет или локална мрежа), за да се илюстрира взаимодействието между компонентите.

**6. Изглед на имплементацията**

Този изглед представя структурните слоеве на системата според MVC архитектурата – контролери, бизнес логика и достъп до данни. Включва компоненти за управление на потребителите, заявки, оферти и модули за работа с базата данни и външни API (Twilio, Google Drive).

**Заключение**

Всеки изглед ще бъде подробно обяснен и подкрепен с диаграми в следващите глави. Те ще осигурят ясна картина на различните аспекти на системата, улеснявайки разбирането и внедряването ѝ.

**Use-Case диаграми**

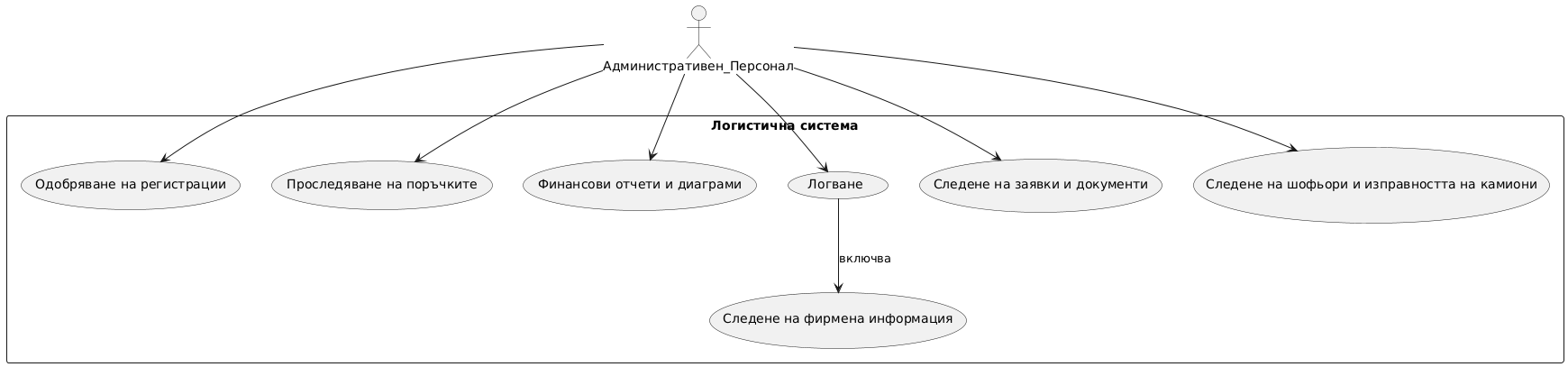
Диаграмата показва основните действия на клиента в логистичната система:

1. Регистрация чрез адрес, БУЛСТАТ, данни за контакт, създаване на нов профил, включва получаване на потребителско име и парола по имейл, ако е одобрена регистрецията.
2. Логване – достъп до системата, включва следене и промяна на фирмени детайли.
3. Подаване на заявка – създаване на заявка за получаване на логистична услуга чрез подаване на данни за товара и преглед на подадените заявки.
4. Получаване на оферта от логистичната фирма вклщчва и преглед на офертите за услугите.
5. Получаване на референтен номер, когато одобрим оферта и се направи доставка – включва следене на доставките и детайли на доставките.
6. Получаване на статистики – достъп до статистически данни за услугите.

**A diagram of a diagram

Description automatically generated with medium confidence**

Диаграмата представя основните функции, достъпни за административния персонал в логистичната система:

1. Одобряване на регистрации – административният персонал може да одобрява нови регистрации на клиенти или други потребители.
2. Проследяване на поръчките – дава възможност на персонала да следи статуса и детайлите на различни поръчки.
3. Финансови отчети и диаграми – предоставя достъп до анализи и визуализации на финансови данни, свързани с дейността на системата.
4. Логване – административният персонал се логва в системата, като това включва и следене на фирмена информация за преглед на данни и настройки, свързани с компанията.
5. Следене на заявки и документи – позволява достъп до заявките на клиенти и свързаната с тях документация.
6. Следене на шофьори и изправността на камиони – дава възможност за следене на данни за шофьорите и състоянието на камионите.

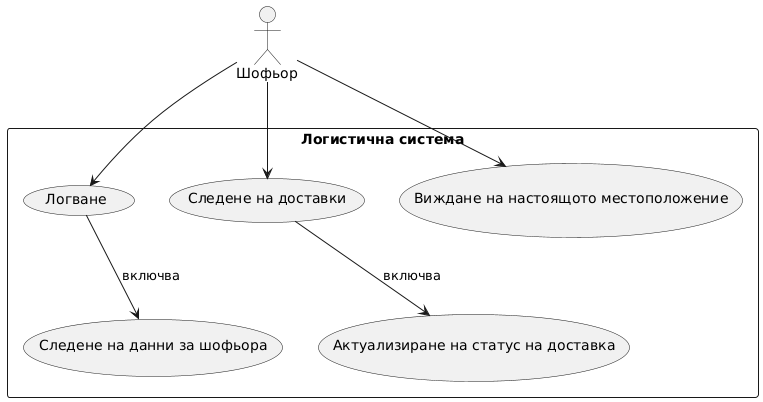
Диаграмата представя функционалностите на логистичната система от гледна точка на счетоводителя:

1. Логване- достъп до системата.
2. Следене на доставки -детайли на доставките, документи, включва добавяне на допълнителни разходи(касови бележки), като те могат да бъдат качени в Google Drive
3. Следене на документи за плащане на доставките - контрол на документите, свързани с плащанията, коетто включва маркиране на сметки като платени
4. Получаване на отчет за неплатени доставки - генериране на отчет за неплатени доставки.



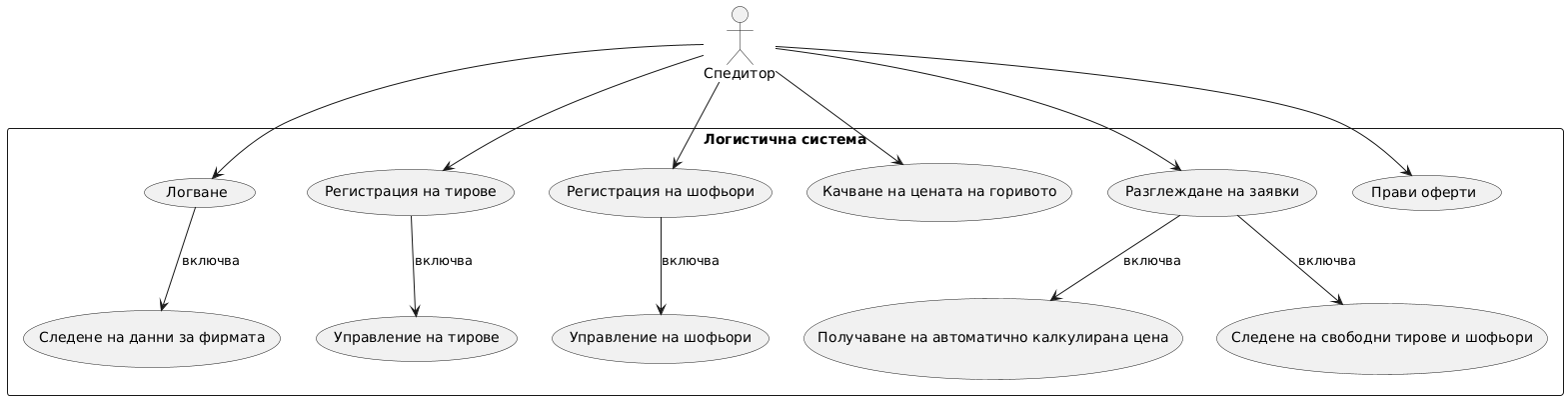
Диаграмата показва взаимодействията на шофьора с логистичната система:

* Логване - достъп до системата, което включва следене на личните данни и шофьорската книжка.
* Следене на доставки -Позволява на шофьора да проследява текущите си доставки и да ктуализира статуса на доставка
* Виждане на настоящото местоположение



Диаграмата представя функционалността на логистичната система от гледна точка на спедитора:

1. Логване -Спедиторът влиза в системата, за да получи достъп до наличните функции. Този процес включва и Следене на данни за фирмата
2. Регистрация на тирове - може да регистрира нови камиони, като за управлението им е предвидена отделна функционалност.
3. Регистрация на шофьори - има възможност да регистрира нови шофьори и да управлява техните данни.
4. Качване на цената на горивото- спедиторът актуализира цените на горивото, за да отрази текущата ситуация на пазара.
5. Разглеждане на заявки - спедиторът преглежда наличните клиентски заявки и може да получи автоматично калкулирана цена за услугата като от това преценя кой тир да избере и да отправи оферта.
6. Прави оферти -може да съставя и управлява оферти на база получените заявки.
7. Следене на свободни тирове и шофьори -спедиторът има функция за наблюдение на наличността на камиони и шофьори, за да оптимизира разпределението на ресурсите.



Диаграмата представя функционалността на логистичната система, свързана с автоматизираните процеси. Тя включва:

1. Изпращане на имейл - автоматизирано изпращане на съобщения до потребители.
2. Известия - генериране на уведомления за важни събития в системата.
3. Изчисляване на цени за оферти - автоматично определяне на цени при създаване на оферти.
4. Визуализация на диаграми - представяне на данни от базата чрез графични диаграми.

A diagram of a diagram

Description automatically generated

**Class diagram**

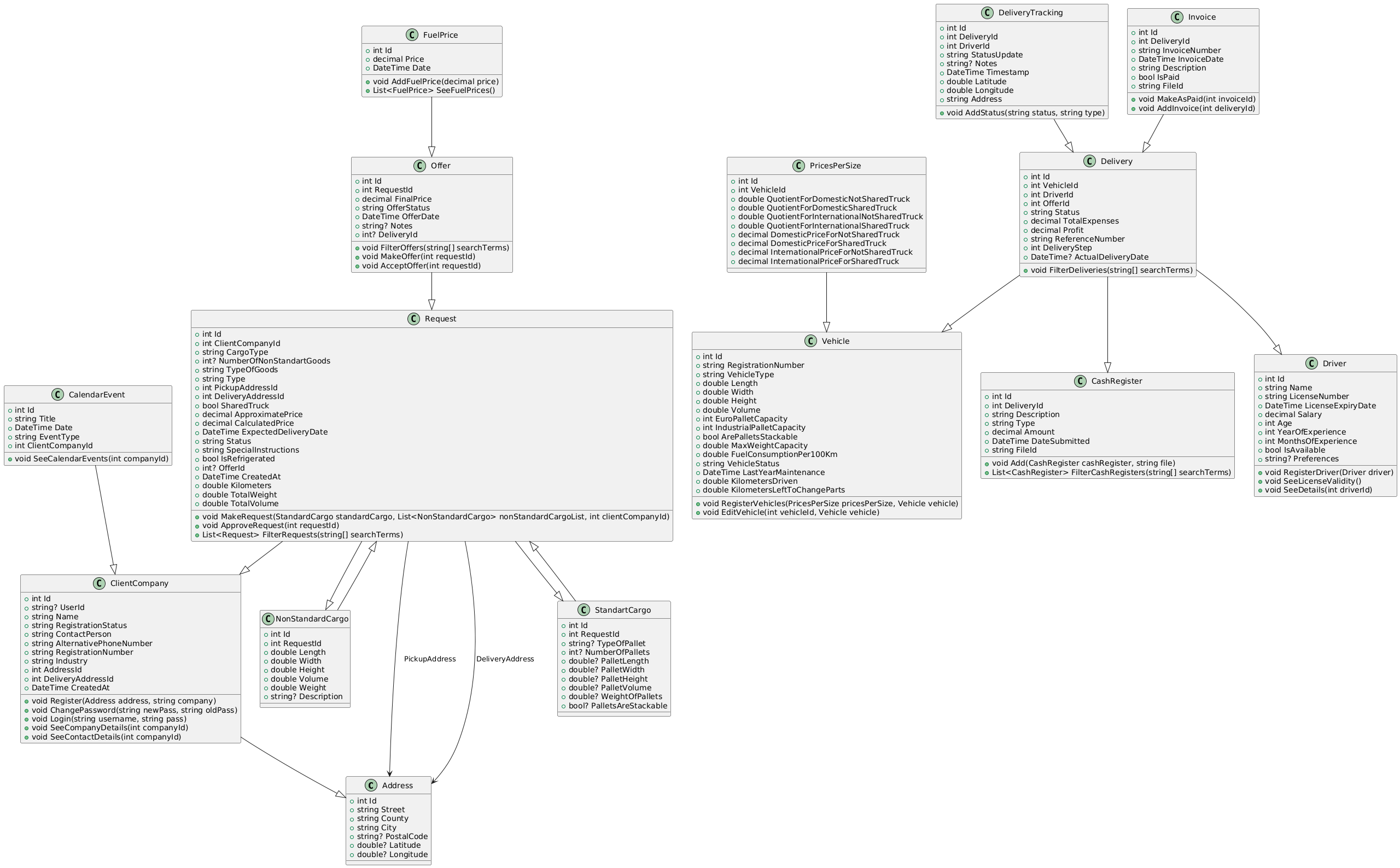
Диаграмата представлява UML (Unified Modeling Language) класова диаграма на логистична система, която обхваща основните класове и техните взаимоотношения.

Класове и техните методи

1. Address
   * Представлява адрес, съдържащ основни данни като улица, окръг, град и пощенски код.
   * Атрибути: Id, Street, County, City, PostalCode, Latitude, Longitude.
2. CalendarEvent
   * Съхранява информация за събития в календара на клиентска компания.
   * Атрибути: Id, Title, Date, EventType, ClientCompanyId.
   * Метод: SeeCalendarEvents(int companyId) - позволява на компанията да види своите събития.
3. CashRegister
   * Представлява касов регистър, свързан с конкретна доставка, и съдържа информация за разходи.
   * Атрибути: Id, DeliveryId, Description, Type, Amount, DateSubmitted, FileId.
   * Методи: Add(CashRegister cashRegister, string file) и FilterCashRegisters(string[] searchTerms) - за добавяне на нови записи и филтриране на съществуващите.
4. ClientCompany
   * Представлява клиентска компания, която ползва услугите на логистичната система.
   * Атрибути: Id, UserId, Name, RegistrationStatus, ContactPerson, AlternativePhoneNumber, RegistrationNumber, Industry, AddressId, DeliveryAddressId, CreatedAt.
   * Методи: Register(Address address, string company), ChangePassword(string newPass, string oldPass), Login(string username, string pass), SeeCompanyDetails(int companyId) и SeeContactDetails(int companyId) - управление на данните на компанията.
5. Delivery
   * Представлява доставка, която включва информация за превозно средство, шофьор, оферта и статус.
   * Атрибути: Id, VehicleId, DriverId, OfferId, Status, TotalExpenses, Profit, ReferenceNumber, DeliveryStep, ActualDeliveryDate.
   * Метод: FilterDeliveries(string[] searchTerms) - за филтриране на доставките.
6. DeliveryTracking
   * Позволява проследяване на статусите на доставките.
   * Атрибути: Id, DeliveryId, DriverId, StatusUpdate, Notes, Timestamp, Latitude, Longitude, Address.
   * Метод: AddStatus(string status, string type) - за добавяне на статус на доставка.
7. Driver
   * Представлява шофьор с информация за лиценз и опит.
   * Атрибути: Id, Name, LicenseNumber, LicenseExpiryDate, Salary, Age, YearOfExperience, MonthsOfExperience, IsAvailable, Preferences.
   * Методи: RegisterDriver(Driver driver) и SeeLicenseValidity().
8. FuelPrice
   * Представлява цените на горивото, използвани при изчисления на офертите.
   * Атрибути: Id, Price, Date.
   * Методи: AddFuelPrice(decimal price) и SeeFuelPrices().
9. Invoice
   * Представлява фактура, свързана с конкретна доставка.
   * Атрибути: Id, DeliveryId, InvoiceNumber, InvoiceDate, Description, IsPaid, FileId.
   * Методи: MakeAsPaid(int invoiceId) и AddInvoice(int deliveryId).
10. NonStandardCargo
    * Представлява нестандартен товар, свързан с конкретна заявка.
    * Атрибути: Id, RequestId, Length, Width, Height, Volume, Weight, Description.
11. Offer
    * Представлява оферта, направена за заявка.
    * Атрибути: Id, RequestId, FinalPrice, OfferStatus, OfferDate, Notes, DeliveryId.
    * Методи: FilterOffers(string[] searchTerms), MakeOffer(int requestId) и AcceptOffer(int requestId).
12. PricesPerSize
    * Съдържа цени за различни видове превозни средства.
    * Атрибути: Id, VehicleId, QuotientForDomesticNotSharedTruck, QuotientForDomesticSharedTruck, QuotientForInternationalNotSharedTruck, QuotientForInternationalSharedTruck, DomesticPriceForNotSharedTruck, DomesticPriceForSharedTruck, InternationalPriceForNotSharedTruck, InternationalPriceForSharedTruck.
13. Request
    * Представлява заявка за транспортна услуга, която включва информация за товара и адресите за вземане и доставка.
    * Атрибути: Id, ClientCompanyId, CargoType, NumberOfNonStandartGoods, TypeOfGoods, PickupAddressId, DeliveryAddressId, SharedTruck, ApproximatePrice, CalculatedPrice, ExpectedDeliveryDate, Status, SpecialInstructions, IsRefrigerated, OfferId, CreatedAt, Kilometers, TotalWeight, TotalVolume.
    * Методи: MakeRequest(StandardCargo standardCargo, List<NonStandardCargo> nonStandardCargoList, int clientCompanyId) и ApproveRequest(int requestId).
14. StandartCargo
    * Представлява стандартен товар, свързан със заявка.
    * Атрибути: Id, RequestId, TypeOfPallet, NumberOfPallets, PalletLength, PalletWidth, PalletHeight, PalletVolume, WeightOfPallets, PalletsAreStackable.
15. Vehicle
    * Представлява превозно средство с информация за техническите му параметри.
    * Атрибути: Id, RegistrationNumber, VehicleType, Length, Width, Height, Volume, EuroPalletCapacity, IndustrialPalletCapacity, ArePalletsStackable, MaxWeightCapacity, FuelConsumptionPer100Km, VehicleStatus, LastYearMaintenance, KilometersDriven, KilometersLeftToChangeParts.
    * Методи: RegisterVehicles(PricesPerSize pricesPerSize, Vehicle vehicle) и EditVehicle(int vehicleId, Vehicle vehicle).

Взаимоотношения между класовете

* Request: свързва ClientCompany, NonStandardCargo и StandardCargo, което показва, че всяка заявка от клиент може да съдържа информация за различни видове товари.
* CashRegister: свързан е с Delivery, което означава, че всяка доставка може да има асоцииран касов регистър.
* Delivery: свързва се с Vehicle, Driver, и Invoice, показвайки как всяка доставка е свързана с конкретно превозно средство, шофьор и фактура.
* FuelPrice: свързан с Offer, което показва, че цената на горивото играе роля в изчисляването на крайната цена на офертата.
* PricesPerSize: свързан с Vehicle, за да отрази различните цени в зависимост от типа на превозното средство.
* CalendarEvent: свързан с ClientCompany, което позволява на компаниите да управляват своите събития.



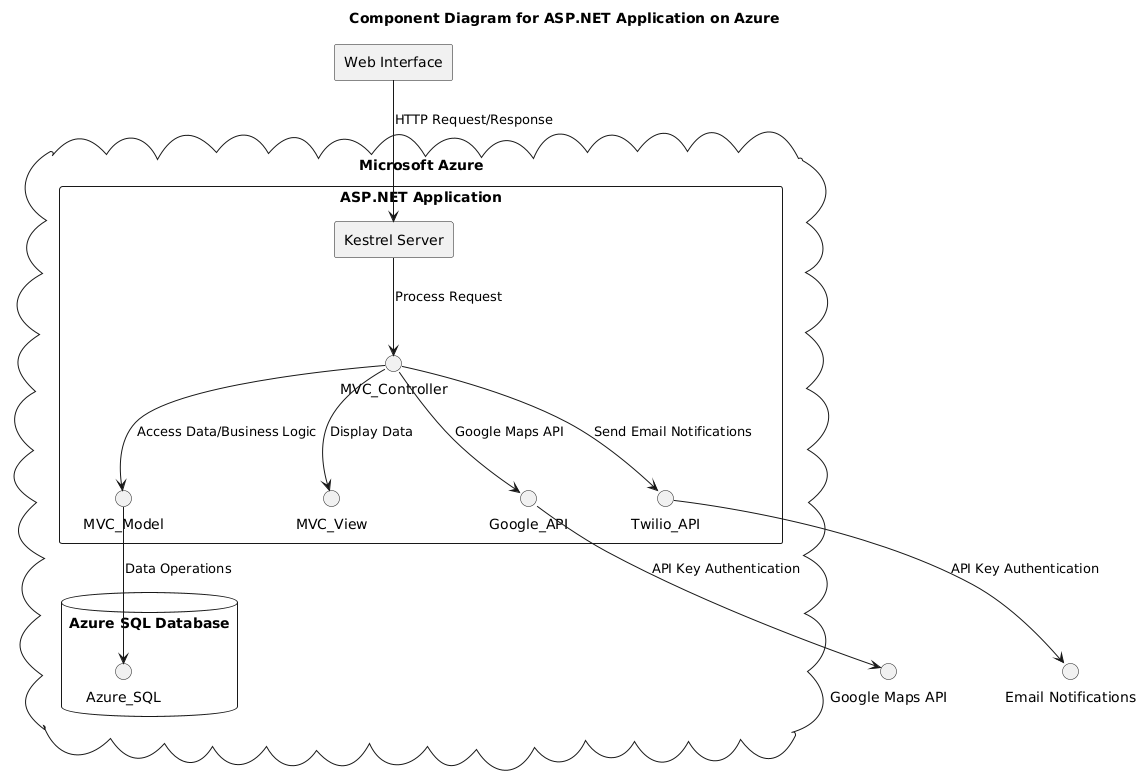
**Component Diagrams**

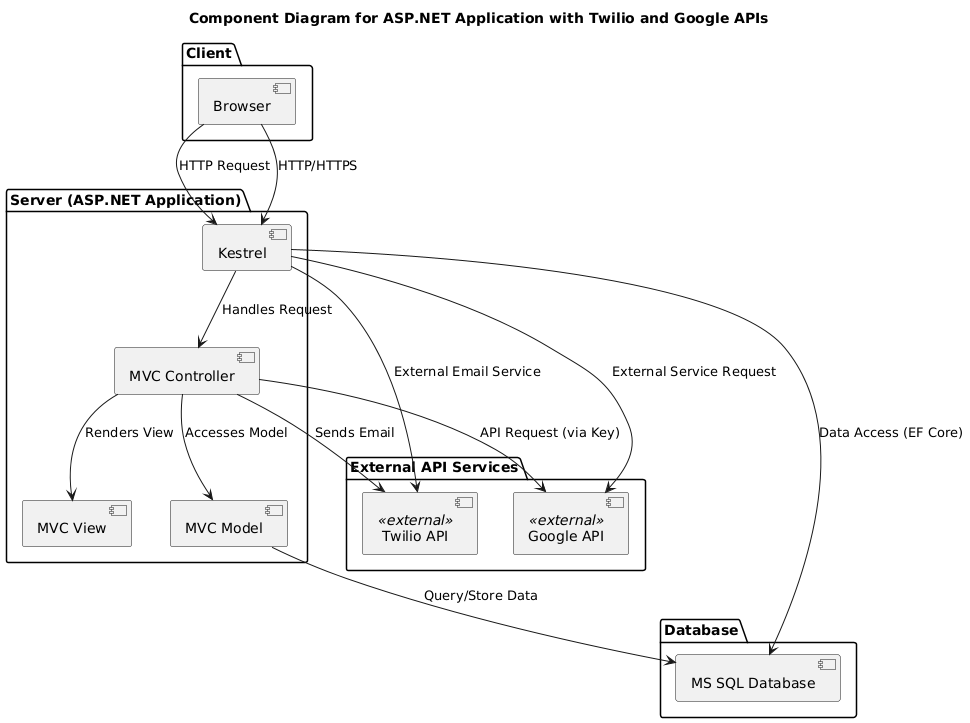
Диаграмата на компонентите показва взаимодействието в ASP.NET приложението:

1. Client (Клиент):
   * Browser (Браузър): Потребителите достъпват приложението чрез браузър.

Cloud Infrastructure (Облачна инфраструктура):

1. Microsoft Azure: Платформата, на която е разположено приложението, осигуряваща гъвкавост и мащабируемост. Azure управлява всички компоненти – Kestrel, базата данни, и свързаните услуги.
2. Server (ASP.NET Приложение):
   * Kestrel: Вграден сървър, който обработва HTTP заявки.
   * MVC Controller (Контролер): Приема заявките и комуникира с моделите и изгледите.
   * MVC Model (Модел): Управлява данните и бизнес логиката.
   * MVC View (Изглед): Рендира отговорите за браузъра.
3. Database (База данни):
   * MS SQL Database: Съхранява информация за клиенти, заявки и поръчки.
4. External API Services (Външни услуги):
   * Google API: Използва се за геолокация и карти.
   * Twilio API: Изпраща автоматични имейли.

Взаимодействия: Браузърът комуникира с Kestrel, който пренасочва заявките към контролерите, а те взаимодействат с модела, изгледа, базата данни и външните API услуги.



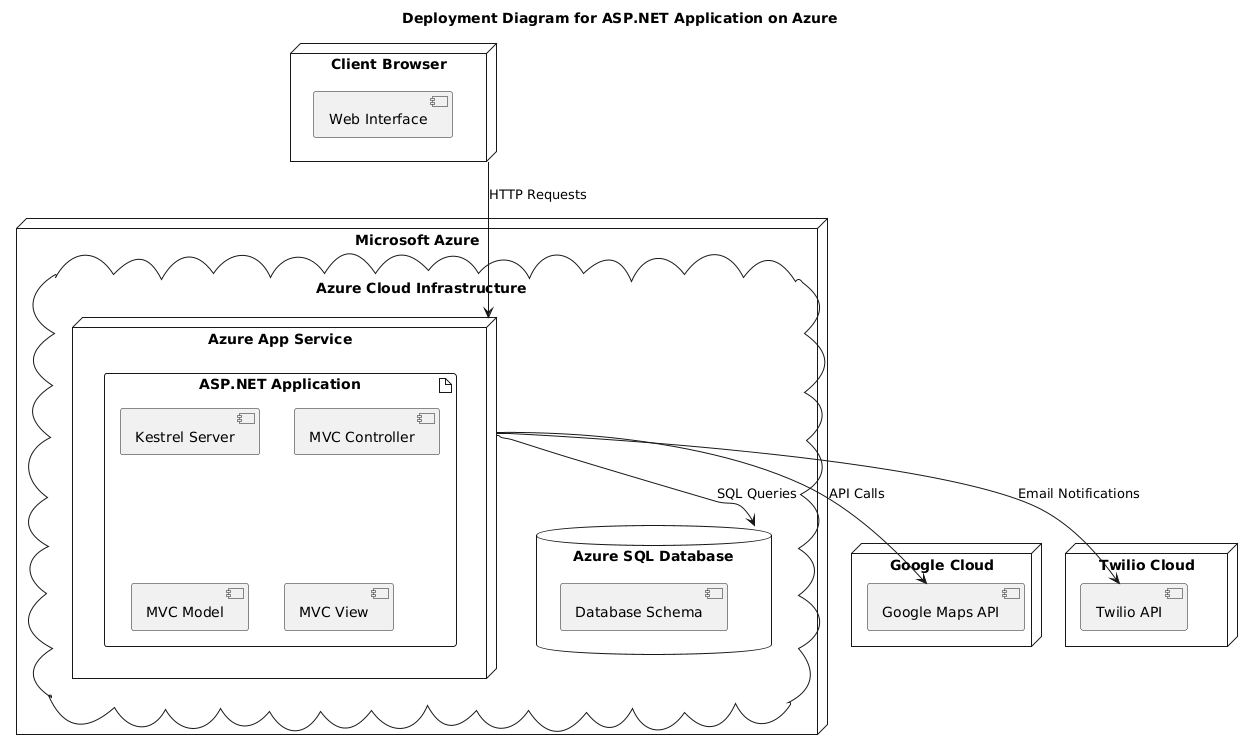
**Deployment Diagrams**

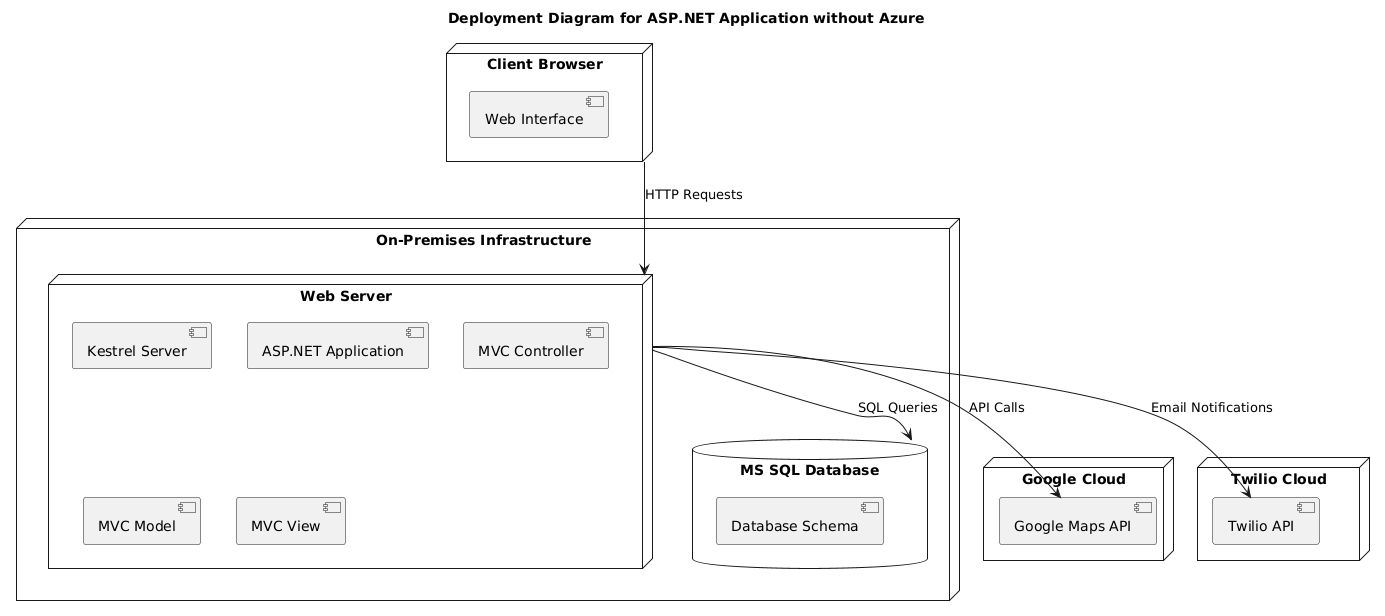
Тази диаграма показва как компонентите на системата са разгърнати и как комуникират помежду си в облачната инфраструктура на Azure.

1. Microsoft Azure: Главната инфраструктура, където е разположено цялото приложение.
   * Azure App Service: Хоства ASP.NET приложението, съдържащо Kestrel сървъра, контролерите, модела и изгледите на приложението.
   * Azure SQL Database: Облачна база данни за съхранение на данните на приложението.
2. Client Browser: Устройството на клиента, което осъществява достъп до приложението чрез уеб интерфейс.
3. Google Maps API и Twilio API: Външни услуги, които приложението използва за геолокация и известия по имейл.

**Описание на връзките:**

* Client Browser → Azure App Service: Клиентите взаимодействат с приложението чрез HTTP заявки.
* Azure App Service → Azure SQL Database: Приложението извършва заявки към базата данни за съхранение и извличане на информация.
* Azure App Service → Google Maps API и Twilio API: Приложението изпраща заявки към външните API-та за специфични функционалности като геолокация и изпращане на имейли.





**Entity Relationship Diagrams**

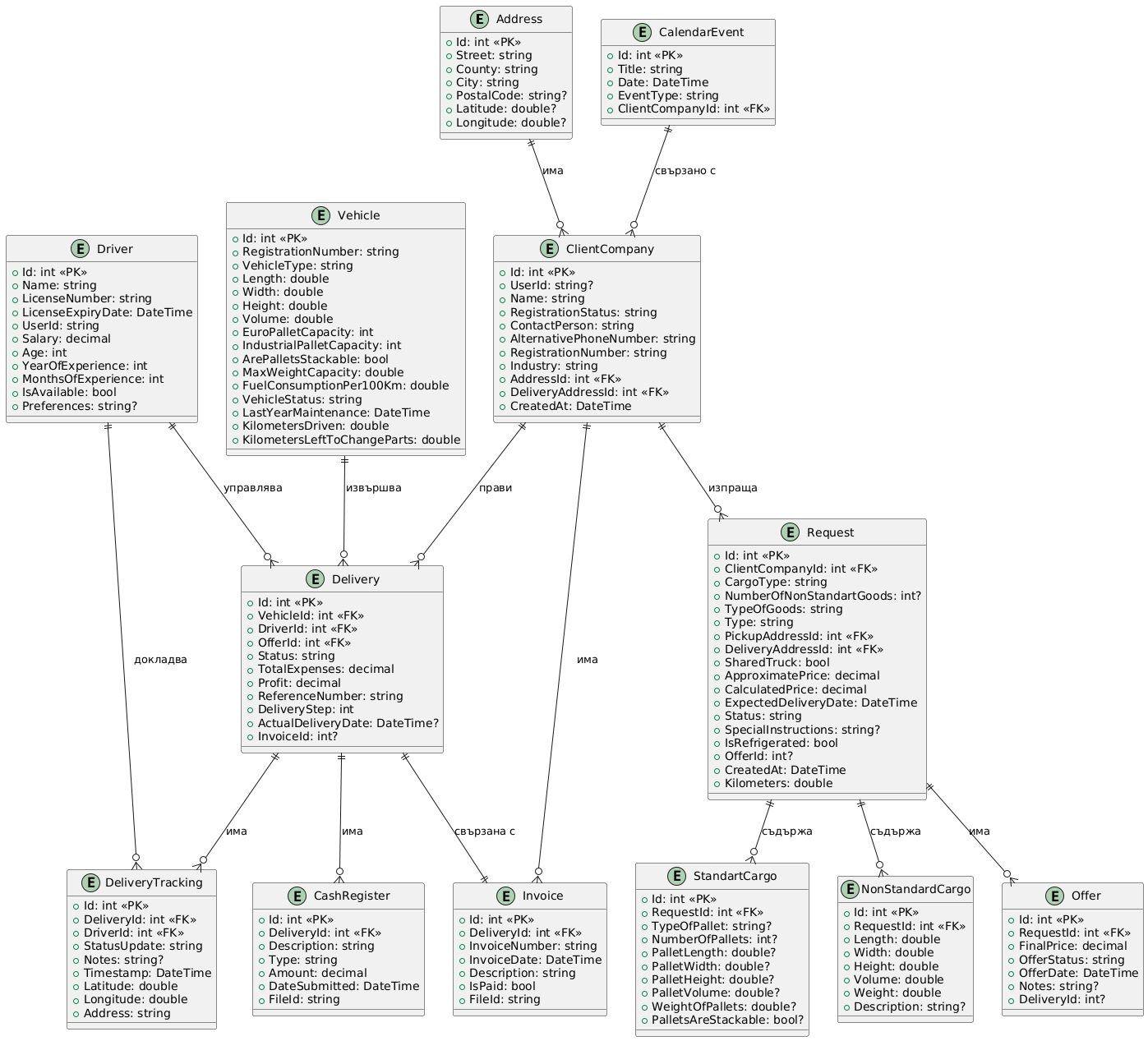
Диаграмата описва структурата на базата данни на логистичната система, включваща основните таблици и техните взаимовръзки.

Класове и техните атрибути

1. Address
   * Атрибути: Id, Street, County, City, PostalCode, Latitude, Longitude.
   * Представлява адрес, който се свързва с клиентските компаниии заявките им за услуги.
2. CalendarEvent
   * Атрибути: Id, Title, Date, EventType, ClientCompanyId.
   * Съхранява информация за събития, свързани с клиентската компания.
3. ClientCompany
   * Атрибути: Id, UserId, Name, RegistrationStatus, ContactPerson, AlternativePhoneNumber, RegistrationNumber, Industry, AddressId, DeliveryAddressId, CreatedAt.
   * Представлява клиентска компания, която ползва логистичните услуги и е свързана с адреси и доставки и евенти.
4. Delivery
   * Атрибути: Id, VehicleId, DriverId, OfferId, Status, TotalExpenses, Profit, ReferenceNumber, DeliveryStep, ActualDeliveryDate, InvoiceId.
   * Представлява доставка, включваща информация за превозно средство, шофьор, оферта и фактура.
5. DeliveryTracking
   * Атрибути: Id, DeliveryId, DriverId, StatusUpdate, Notes, Timestamp, Latitude, Longitude, Address.
   * Позволява проследяване на доставките и свързаната информация за статуси.
6. Driver
   * Атрибути: Id, Name, LicenseNumber, LicenseExpiryDate, UserId, Salary, Age, YearOfExperience, MonthsOfExperience, IsAvailable, Preferences.
   * Представлява шофьор с информация за лиценза и опита.
7. CashRegister
   * Атрибути: Id, DeliveryId, Description, Type, Amount, DateSubmitted, FileId.
   * Представлява касов регистър, свързан с доставка.
8. Invoice
   * Атрибути: Id, DeliveryId, InvoiceNumber, InvoiceDate, Description, IsPaid, FileId.
   * Представлява фактура, свързана с доставка.
9. NonStandardCargo
   * Атрибути: Id, RequestId, Length, Width, Height, Volume, Weight, Description.
   * Представлява нестандартен товар, свързан със заявка.
10. Offer
    * Атрибути: Id, RequestId, FinalPrice, OfferStatus, OfferDate, Notes, DeliveryId.
    * Представлява оферта, направена за заявка.
11. Request
    * Атрибути: Id, ClientCompanyId, CargoType, NumberOfNonStandartGoods, TypeOfGoods, Type, PickupAddressId, DeliveryAddressId, SharedTruck, ApproximatePrice, CalculatedPrice, ExpectedDeliveryDate, Status, SpecialInstructions, IsRefrigerated, OfferId, CreatedAt, Kilometers.
    * Представлява заявка за транспортна услуга.
12. StandartCargo
    * Атрибути: Id, RequestId, TypeOfPallet, NumberOfPallets, PalletLength, PalletWidth, PalletHeight, PalletVolume, WeightOfPallets, PalletsAreStackable.
    * Представлява стандартен товар, свързан със заявка.
13. Vehicle
    * Атрибути: Id, RegistrationNumber, VehicleType, Length, Width, Height, Volume, EuroPalletCapacity, IndustrialPalletCapacity, ArePalletsStackable, MaxWeightCapacity, FuelConsumptionPer100Km, VehicleStatus, LastYearMaintenance, KilometersDriven, KilometersLeftToChangeParts.
    * Представлява превозно средство с технически параметри.

**Взаимоотношения между класовете**

1. Address - Свързва се с ClientCompany (има)  
   Всяка клиентска компания има адрес.
2. ClientCompany - Свързва се с:
   * Delivery - клиентската компания може да прави доставки.
   * Request - клиентската компания може да изпраща заявки.
   * Invoice - клиентската компания има фактури.
   * CalendarEvent – клиентската компания има евенти
3. Request - Свързва се с:
   * NonStandardCargo заявката може да съдържа нестандартни товари.
   * StandartCargo -заявката може да съдържа стандартни товари.
   * Offer -заявката може да има оферта.
4. Delivery - Свързва се с:
   * DeliveryTracking - доставка може да има записи за проследяване.
   * CashRegister- доставка може да има свързан касов регистър.
   * Invoice -всяка доставка е свързана с фактура.
5. Vehicle - Свързва се с Delivery   
   Всяка доставка е извършвана от превозно средство.
6. Driver - Свързва се с:
   * Delivery - шофьорът управлява доставките.
   * DeliveryTracking - шотчита статуси на доставките.
7. CalendarEvent - Свързва се с ClientCompany   
   Събитията в календара са свързани с клиентските компании.



**Package Diagram**

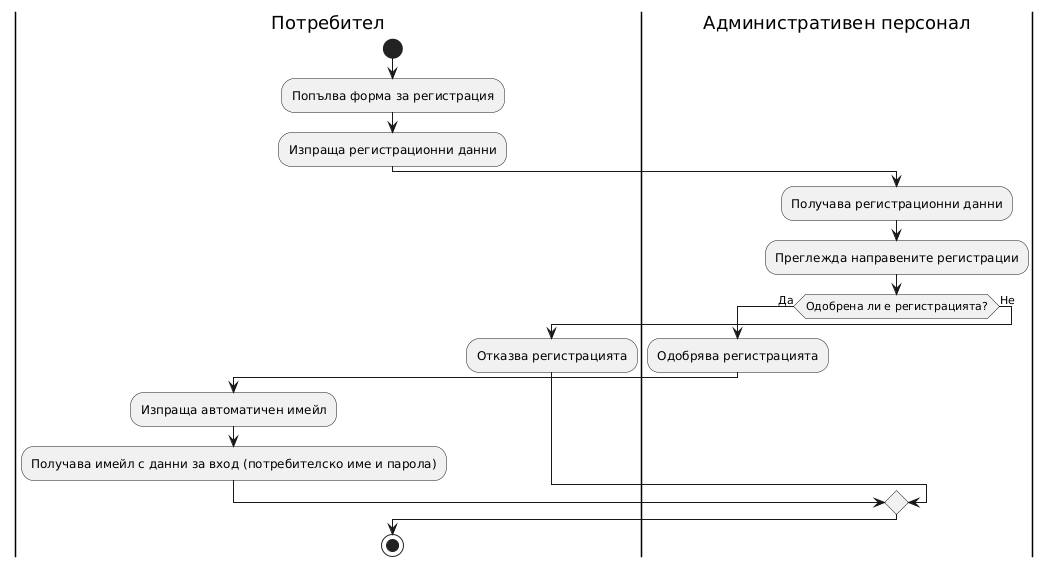
Тази диаграма представя основната архитектура на вашето приложение и как различните слоеве взаимодействат помежду си.

1. Presentation Layer:
   * Controllers: Отговарят за обработката на входящите заявки от потребителите и взаимодействието с бизнес логиката.
   * ViewModels: Служат за предаване на данни между контролерите и представянето на потребителския интерфейс.
2. Business Logic Layer:
   * Services: Съдържат бизнес логиката на приложението и взаимодействат с репозиториите, за да манипулират данни.
3. Data Access Layer:
   * Repositories: Управляват взаимодействието с базата данни, предоставяйки методи за извличане и записване на данни.
   * Entities: Представляват моделите на данните, които се съхраняват в базата данни.
   * DbContext: Управлява връзката между приложението и базата данни.

**Activity Diagrams**

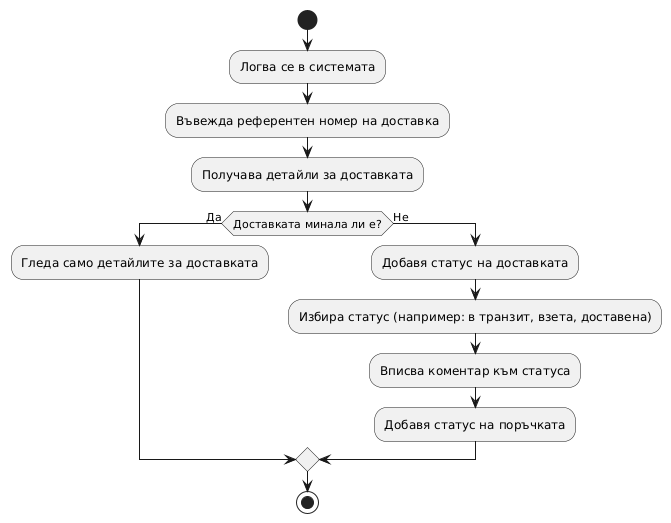
Тази диаграма илюстрира основните стъпки и решения, които се вземат по време на процеса на регистрация и одобрение.

1. Потребител: Започва с попълване на форма за регистрация и изпращане на регистрационните данни.
2. Административен персонал: Получава регистрационните данни и преглежда направените регистрации.
3. Условие: Проверява дали регистрацията е одобрена или не.
   * Ако е одобрена:
     + Изпраща се автоматичен имейл до потребителя с данни за вход (потребителско име и парола).
   * Ако не е одобрена:
     + Регистрацията се отказва.
4. Процесът приключва.



Тази диаграма илюстрира стъпките, които шофьорът предприема след логването, и действията, които може да извърши в зависимост от състоянието на доставката.

1. Шофьор: Процесът започва, когато шофьорът се логва в системата.
2. Въвеждане на референтен номер: Шофьорът въвежда референтен номер на доставка, след което системата получава и показва детайлите за доставката.
3. Условие: Проверява се дали доставката е минала.
   * Ако е минала:
     + Шофьорът вижда само детайлите за доставката.
   * Ако не е минала:
     + Шофьорът може да добави статус на доставката.
     + Избира статус (например: в транзит, взета, доставена).
     + Вписва коментар към статуса и добавя статус на поръчката.
4. Процесът приключва.

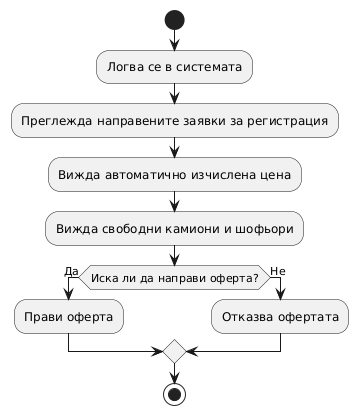


Тази диаграма показва стъпките, които счетоводителят предприема след логването, в зависимост от статуса на доставките и действията, които може да извърши.

1. Счетоводител: Процесът започва, когато счетоводителят се логва в системата.
2. Филтриране на доставките: Счетоводителят филтрира доставките, за да види списък с тях.
3. Условие: Проверява се дали доставките са завършили.
   * Ако са завършили:
     + Счетоводителят добавя касови бележки, които са допълнителни разходи.
     + Вписва детайли за касовата бележка.
     + Качва сканираните бележки като файлове в Google Drive.
   * Ако не са завършили:
     + Счетоводителят вижда само детайлите за настоящите доставки.
4. Процесът приключва.

Тази диаграма показва стъпките, които спедиторът предприема след логването, включително как взима решение относно офертите.

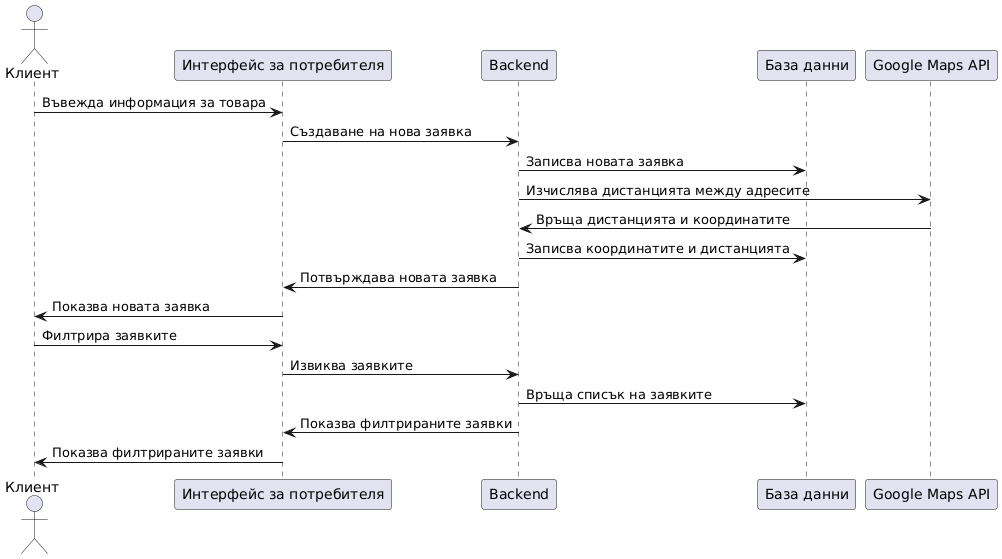
1. Спедитор: Процесът започва, когато спедиторът се логва в системата.
2. Преглеждане на заявки за регистрация: Спедиторът преглежда заявките за регистрация, които са направени от клиентите.
3. Автоматично изчислена цена: Спедиторът вижда автоматично изчислената цена за предлаганата услуга.
4. Свободни камиони и шофьори: Спедиторът преглежда наличността на свободни камиони и шофьори.
5. Условие: Проверява се дали спедиторът иска да направи оферта.
   * Ако иска:
     + Спедиторът прави оферта.
   * Ако не иска:
     + Спедиторът отказва офертата.
6. Процесът приключва.



**Communication Diagrams**

Тази диаграма илюстрира процеса на правене на заявка от клиентите, включително взаимодействието с базата данни и Google Maps API.

1. Клиент: Клиентът започва, като въвежда информация за товара си в потребителския интерфейс.
2. Интерфейс за потребителя (UI): Получава информацията и изпраща заявка за създаване на нова заявка към бекенда.
3. Backend (BE): Бекендът записва новата заявка в базата данни.
4. Google Maps API (GM): Бекендът извиква Google Maps API, за да изчисли дистанцията между адреса за вземане и адреса за доставка.
5. Google Maps API (GM) -> Backend (BE): API-то връща изчислената дистанция и координатите на адресите.
6. Backend (BE): Бекендът записва координатите и дистанцията в базата данни.
7. Backend (BE): Потвърдата на новата заявка се изпраща обратно към потребителския интерфейс.
8. Интерфейс за потребителя (UI): Показва новата направена заявка на клиента.
9. Клиент: Клиентът може да избере да филтрира заявките си.
10. Интерфейс за потребителя (UI): Извиква заявките от бекенда.
11. Backend (BE): Извиква базата данни за списък на заявките.
12. Backend (BE): Връща списъка на заявките.
13. Интерфейс за потребителя (UI): Показва филтрираните заявки на клиента.

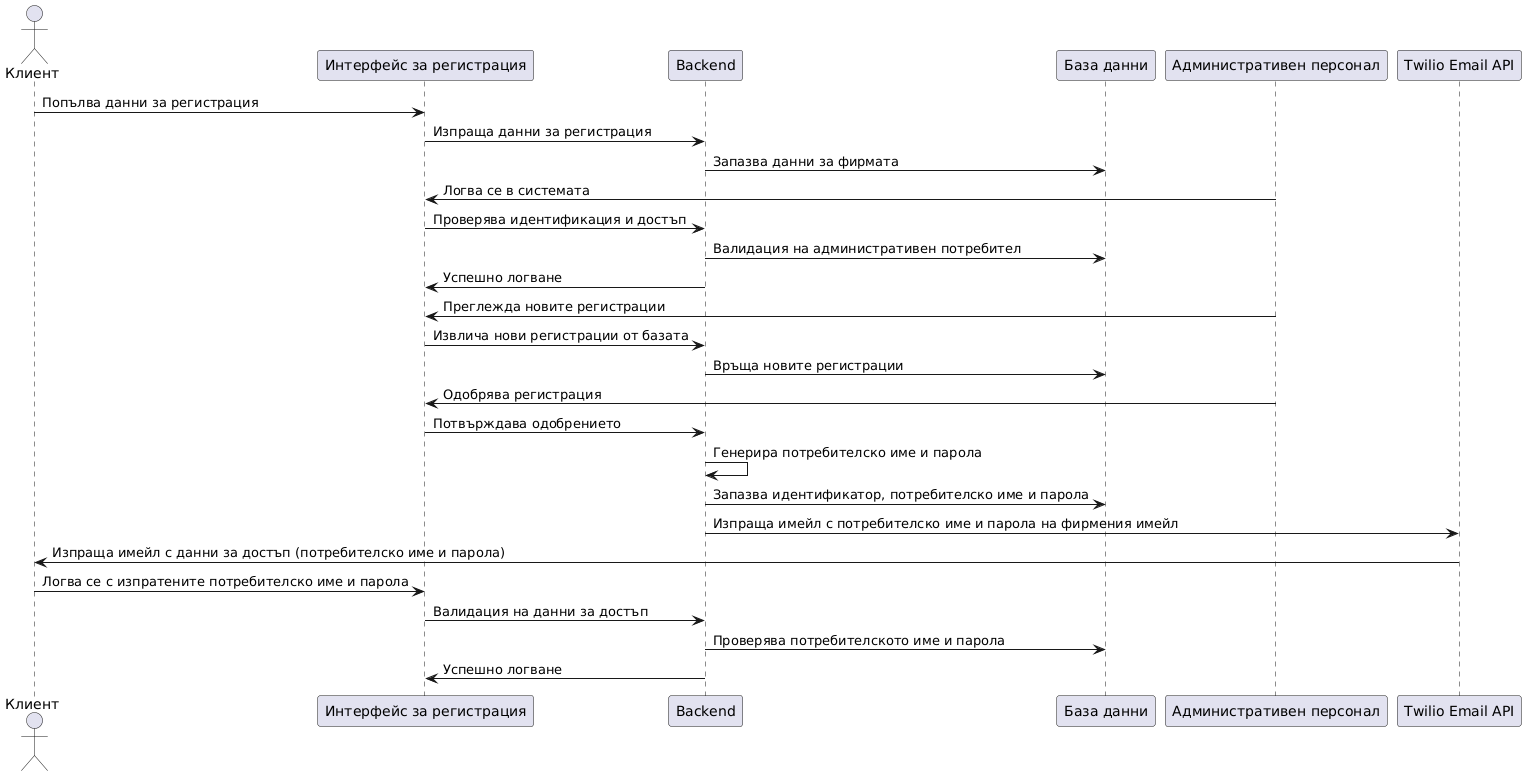


Тази диаграма илюстрира процеса на логване на спедитор в системата, извличане на заявките за транспорт, направени от клиентите, получаване на цена и калкулирано разстояние и маршрут и отправяне на оферта.

1. Спедитор: Логва се в системата чрез интерфейса за спедитори.
2. Интерфейс за спедитор (UI): Проверява данните за достъп и ги изпраща към бекенда за валидация.
3. Backend (BE): Потвърждава логването и връща информацията за спедитора.
4. Спедитор: Преглежда новите заявки за транспорт.
5. Backend (BE): Извлича от базата новите заявки и техните детайли, като километри, адреси, и дали заявката е за споделен камион.
6. Backend (BE): Търси налични камиони и шофьори, подходящи за заявките.
7. Backend (BE): Ако заявката е за споделен камион, извиква Google Maps API, за да изчисли маршрута и километрите между адресите на обединените заявки.
8. Backend (BE): Калкулира цената за офертата спрямо избрания камион, цената за размер, текущата цена на горивото и километрите.
9. Спедитор: Избира камион и шофьор за всяка заявка.
10. Backend (BE): Записва избора на камион и шофьор за заявката и създава оферта за клиента с калкулираната цена.
11. UI: Потвърждава на спедитора изпращането на офертата и уведомява клиента за получената оферта.

Тази диаграма илюстрира процеса на регситрация на клиентска фирма, одобрение на регистрацията от административния персонал и пращането на имейл чрез Twillio на информация за логването на клиентите.

1. Клиент: Попълва данни за регистрация чрез интерфейса за регистрация.
2. Backend (BE): Получава и записва регистрационните данни на клиента в базата данни.
3. Административен персонал: Логва се в системата, която валидира идентификацията и достъпа му.
4. Административен персонал: Преглежда новите регистрации в системата и избира да одобри определена регистрация.
5. Backend (BE): След одобрението генерира потребителско име и парола за клиента и запазва тези данни в базата данни.
6. Twilio Email API: Backend използва Twilio, за да изпрати имейл на клиента с новите данни за достъп.
7. Клиент: Получава имейл с потребителско име и парола и може да се логне в системата.



Тази диаграма илюстрира процеса на логване на счетоводител, търсенето на поръчка по референтния й номер, добавянето на касов бон като допълнителен разход, възникнал по време на доставката и добавянето му като файл на сканирания касов бон в Google Drive

1. Счетоводителят се логва в системата, като въвежда своите данни за достъп.
2. Backend проверява данните за достъп чрез базата данни и потвърждава логина.
3. Счетоводителят въвежда референтен номер на доставката, за да провери свързаните разходи.
4. Backend извлича от базата данни детайлите на доставката и съществуващите касови бележки, след което ги показва на счетоводителя.
5. Счетоводителят добавя нов касов регистър като въвежда описание, сума и тип разход.
6. Счетоводителят качва сканиран файл на касов бон, който Backend изпраща към Google Drive.
7. Google Drive връща линк към качения файл, който се запазва заедно с новия касов регистър в базата данни.
8. Backend потвърждава успешно добавения касов регистър.

**Забележки**

След представяне на различни функционалности от нашия софтуер, следва да започнем работа по реализацията на проекта. Това е начален замисъл на системата и основни части от нея, които подлежат на допълнително дискотируане с групата. Нефункционалните изисквания като достъпност, мониторинг, хостване и други са част от проекта, които подлежат на допълнителни обсъждания. Поради липса на знания в групата за технлогии, които могат да помогнат, а и да бъдат безплатни, сме ги оставили за най-накрая и ще е интересно предизвикателнство да можем да ги включим в ппроекта. Друг аспект от проекта е разпределението на времето, което е от изкючителна важност. Надяваме се да изградим един наистина интересен и полезен софтуер като съчетаем техноологии, които познаваме и нови които да включим.