

Технически Университет гр.Варна

Факултет по изчислителна техника и автоматизация

Специалност “Софтуерни и интернет технологии”

Курсов проект

**Коли под наем**

*Виктор Красенов Драгоев - 17621640*

**Коли под наем**

Да се разработи информационна система, предоставяща възможност за отдаване на коли под наем.

Системата поддържа вида потребители: администратор и оператори с различни роли за достъп до функционалностите в системата.

Операции за работа с потребители:

* Създаване на фирма за отдаване на коли под наем от администратор;
* Създаване на оператори отдаващи коли от администратор;
* Регистриране на клиенти от оператори

Системата поддържа операции за отдаване на коли под наем :

* Регистрация на автомобили (класове - луксозна, семейна, градска , категории - седан, сув, комби, характеристики , снимки, пушач, непушач);
* Отдаване на автомобил на клиент с запазване на текущото състояние автомобила под формата на опис протокол;
* Връщане на автомобил и статус на връщането (безпроблемно, с възникнали проблеми - нов опис протокол за състоянието на автомобила);
* Формиране на дължимата цена спрямо броя дни на наемане, изминати километри и евентуални щети по автомобила (различните класи автомобили имат различна цена за ден и за километър);

Системата поддържа Справки по произволен период за:

* Налични автомобили;
* История за отдаване на автомобили;
* По оператори и тяхната работа;
* Рейтинговане на клиенти;
* Статистики на отдадените автомобили по зададени филтри - по класа, модел, тип и др.

Системата поддържа Известия за :

* Изтичащ или изтекъл период на отдаване;
* Рисков клиент;

Кола под наем е превозно средство, което може да се използва временно срещу заплащане за желания период. Наемането на кола под наем помага на хората да се придвижват до желаните им места въпреки факта, че не разполагат с лично превозно средство или са далеч от него .Лицето което се нуждае от кола трябва да се свържи с компания за коли под наем и да сключи договор съвместно ползването и. За тази цел клиентът трябва да предостави информация като период на наемане, клас и категория на автмобила заедно с валидна книжка.

Системата за коли под наем предложена от нас е напълно функционална и улеснява работата на операторите.

* Лесна за използване
* Пести много време и пари
* Наблюдението на автомобилите, операторите и бизнеса става лесно.
* Замества ползването на хартия, което прави системата екологична
* Повишава ефективността на управлението при предлагането на качествени услуги на клиентите.

***Структура на проекта***

**Админ**

* Може да се логне в системата като администратор, което му позволява да ползва допълнителни функционалностти.
* Може да създава фирми
* Може да създава оператори и да следи тяхната дейност
* Може да създава категории
* Може да създава класове
* Може да прави регистрация на автомобил
* Може да прави регистрация на клиент
* Може да прави справки за налични автомобили
* Може да прави справки за отдавани автомобили
* Може да прави справки за рейтинга на клиентите
* Може да прави статистика за отдадените автомобили

**Оператор**

* Може да създава наемания на автомобил
* Може да създава връщания на автомобил
* Може да преглежда създадени описи на автомобили
* Може да регистрира автомобил
* Може да регистрира клиент
* Може да прави справки за налични автомобили
* Може да прави справки за отдавани автомобили
* Може да прави справки за рейтинг на клиентите
* Може да прави статистика за отдадените автомобили по класове и спецификации

Няма ограничение в операционната система,на която ще работи системата за наемане на автомобили. Като базата данни ще работи на сървър, който трябва да е винаги онлайн.

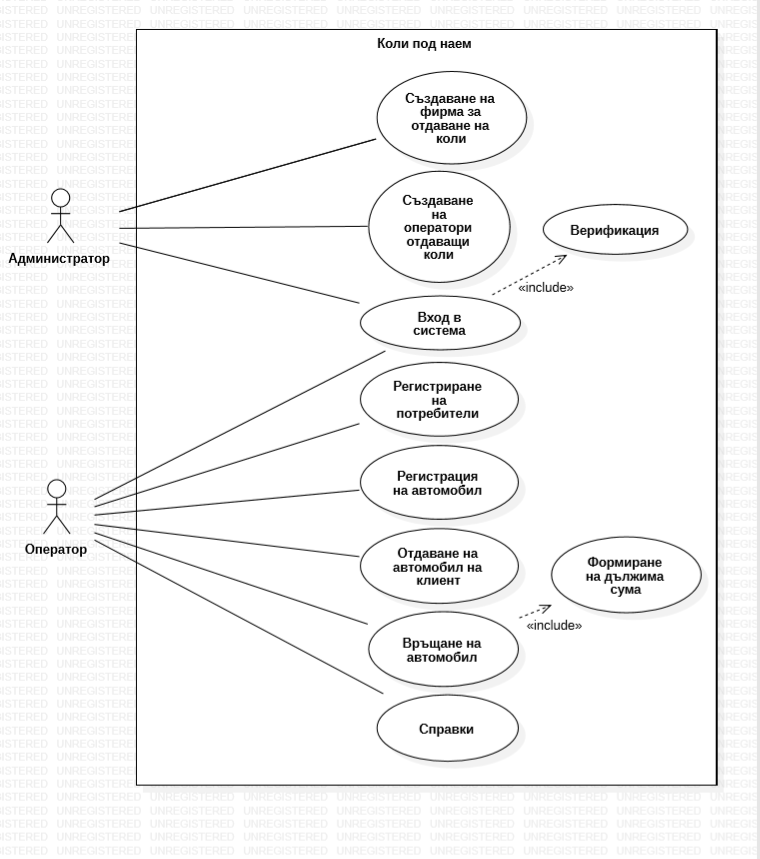
Има функционалности , които системата трябва да предостави. Например как ще реагира на дадена входяща информация или как системата трябва да се държи в конкретна ситуация. Затова трябва да разработим тези функционалности в продукта си за да може потребителите да изпълнят задачите си.

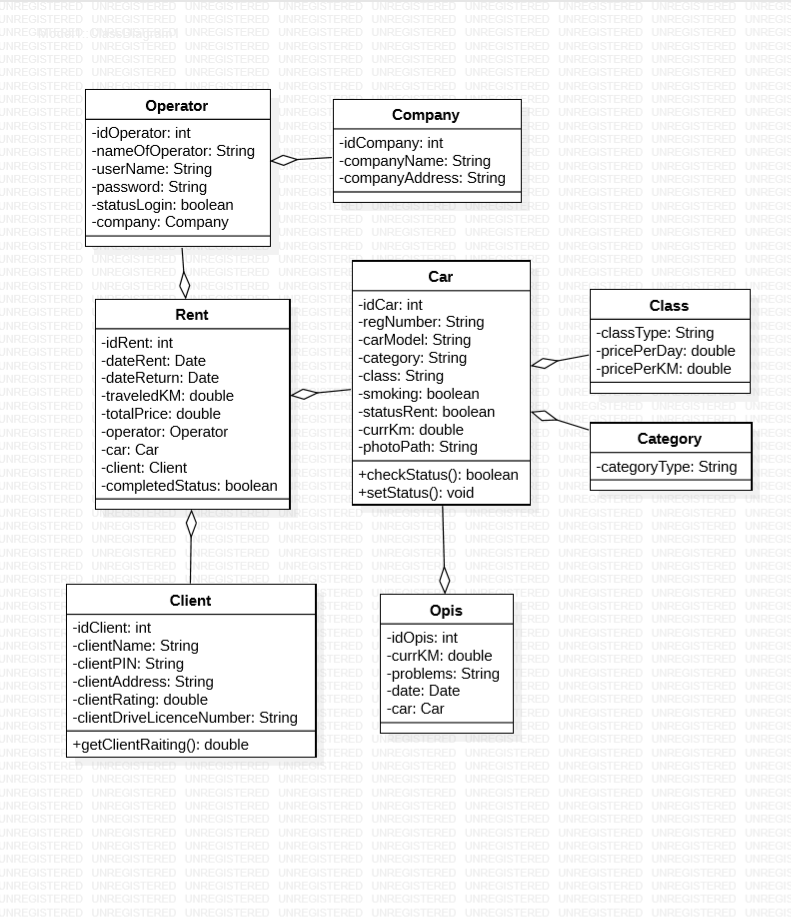
**Функционалности за наемане**

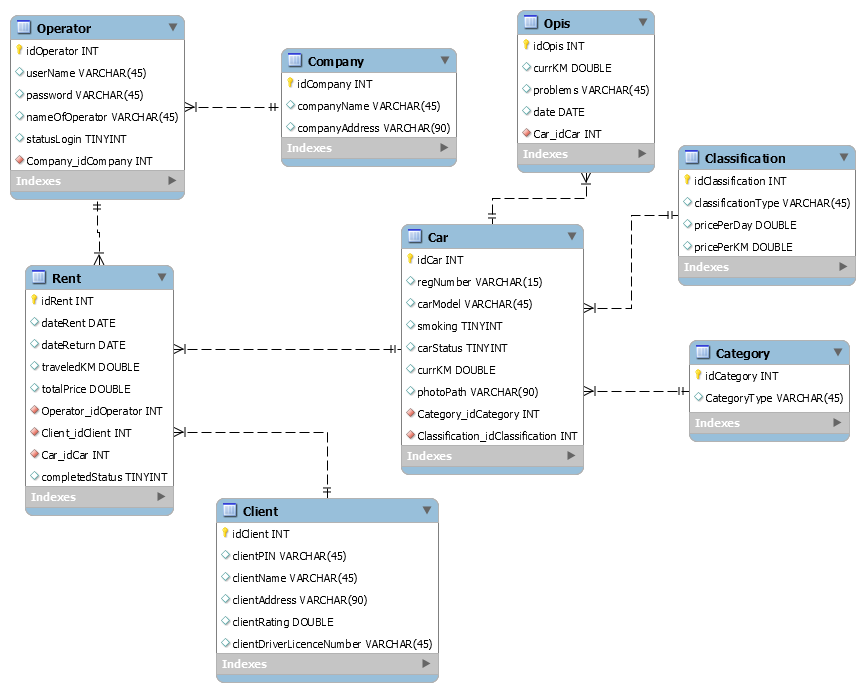
* Системата трябва да позволява регистрация на клиенти.
* Системата трябва да позволява на клиента да види подробно описание за конкретен автомобил.
* Системата трябва да показва само наличните за наемане автомобили.
* Системата трябва да представя опция за разширено търсене, за да ограничи търсенето на автомобил до  
  конкретни категории.
* Системата трябва да предоставя списък на наличните автомобили по време на наемане
* Системата трябва да може да показва успешно извършено наемане.
* Системата трябва да позволява на операторите да регистрират клиенти в списъка за наемане
* Системата трябва да може да запише всички промени, направени в списъка с наеманията на клиенти.
* Системата позволява на оператора да избира клиент за наем на кoла според конкретна категория за търсене.
* Системата трябва да позволява на операторите да регистрират нови автомобили.
* Системата трябва да позволява на оператора да проследява клиента по рейтинг в списък.
* Системата трябва да позволява на оператора да показва списък с всички налични автомобили за период.
* Системата трябва да позволява на оператора да показва всички наети автомобили в период.

**Вход в системата**

* Системата трябва да позволи на операторите да влизат в нея, използвайки потребителското си име и парола
* Системата трябва да позволява на администратора да влиза , използвайки потребителското си име и парола.
* Системата позволява на администратора да създаде нов потребителски акаунт (Оператор).
* Системата трябва да позволява на операторите да излизат.
* Системата трябва да позволява на администратора да излиза.
* Системата трябва да забрани логването на един потребител повече от един път.
* Системата трябва да бъде оборудвана с проверка за коректни данни.







За реализирането на проекта RentACar сме избрали да ползваме продукта MySQL за изработката на базата от данни.Таблиците са свързани с необходимите връзки за функционирането на системата.

Таблица Operator съдържа: потребителско име , парола , име на оператор, статус за вход в системата и фирма.

Таблица Company съдържа: име на фирма и адрес на фирма.

Таблица Opis съдържа: текущи километри , възникнали проблеми ,дата на създаване и автомобил.

Таблица Classification съдържа: типа на класа, цена за ден , цена за километър .

Таблица Category съдържа: тип категория.

Таблица Car съдържа: регистрационен номер , модел на автомобил , статус за пушачи , статус за наемане , текущи километри , местоположение на снимка , категория и клас.

Таблица Client съдържа: ЕГН на клиент , име , адрес , рейтинг и номер на шофьорска книжка.

Таблица Rent съдържа: дата на наемане , дата за връщане , изминати километри , цена за наемане , оператор , клиент , кола и статус.

Правила:

Всеки оператор има една фирма.

Всяко наемане има един оператор, един клиент и една кола.

Всяка кола има категория и клас.

Всеки опис има една кола.

За слоя на работа с базата данни избрахме да ползваме Hibernate, който представлява обектно-релационно средство за картографиране на езика Java. В програмата всеки клас има атрибути, които са картографирани с Hibernate. Всеки атрибут на съответните класове е свързан с колона в таблица от базата от данни. За някой от тях сме използвали анотация от типа Many-to-one.

@ManyToOne(cascade = CascadeType.***ALL***)

**public** Company getCompany() {

**return** company;

}

За уникалните ключове е използвана анотация @Id и @GeneratedValue за автоматично генериране на стойност за ключа.

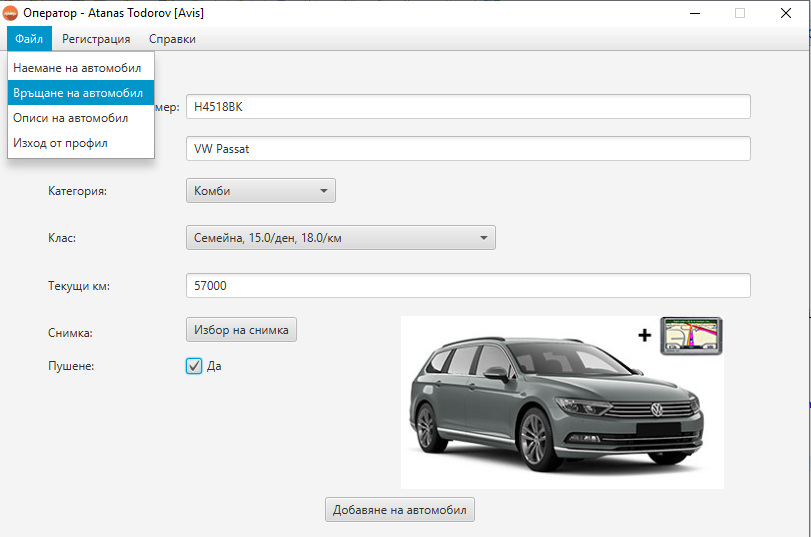
@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

@Column(name="idCar")

Използван е клас HibernateUtil за получаване на SessionFactory за изграждане на сесии. За всяка заявка е използвана отделна сесия. По този начин се подобрява производителността на програмата.

Session session = rentacar.HibernateUtil.*getSessionFactory*().openSession();



За реализация на графичния интерфейс е използван JavaFX чрез SceneBuilder. За визуализирането на данни са използвани: TableVew, ImageView, Label. За предоставянето на избор: ComboBox и ChoiceBox. За входни данни TextField. Създаване на статистика: BarChar. Основното меню в програмата е реализирано с MenuBar. Създадени са различни сцени, които се отварят в главния прозорец (stage).

За реализация на модула за регистриране на събития използвахме log4J. Шаблонът за записване на събития е запсан в log4J.properties.

* За запис на събитие - вход в системата:

**final** **static** Logger ***logger*** = Logger.*getLogger*(LoginControler.**class**);

***logger***.info("Operator"+logedOperator.getUserName()+"-"

+logedOperator.getNameOfOperator()+" logged in.");

* За запис на събитие - наемане на автомобил:

***logger***.info("Operator "+operator.getUserName()+" rented "+car1+" to a client "+choosenClient.getClientName()+choosenClient.getClientPIN());

* За запис на събитие - връщане на автомобил:

***logger***.info("Client "+rent.getClient().getClientName()+" "+rent.getClient().getClientPIN()+" returned car "+rentedCar+" to operator "+operator.getUserName());

**Бизнес логика**

При наемане на автомобил операторът избира клиент и наличен автомобил от списък (с възможност да търсене по категории), задава дата на връщане на автомобила и запазва данните в базата от данни. При връщане на автомобил се избира съответният нает автомобил на клиента, въвеждат се текущите километри и евентуални щети по автомобила и се извежда информация за наемането (клиент, кола, изминати километри и цена). Ако колата е върната с проблем рейтингът на клиента спада с 3 точки. При закъснение за връщане на автомобил рейтингът спада с 5 точки.

**Начини за подобряване на работата**

Системата може да се подобри , като направим приложението да работи чрез интернет с възможност за плащане чрез банков път. Така редовните клиенти ще наемат коли без да се налага да ходят до офис на фирмата и всички останали ще могат да разглеждат услугите, които предлагаме.

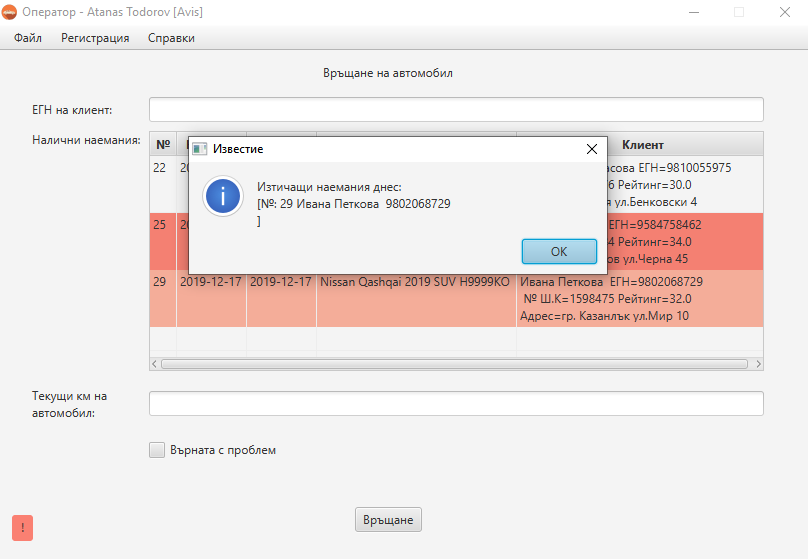
Разработване на опция за отказване/отменяне на наемане , така в случай на възникнали проблеми , ще може да се откаже дадено наемане.

Добавяне на функция към администраторския профил , която ще прави промени върху цените на различните модули.

Добавяне на нови езици , като така ще привлечем повече клиенти.  
 Добавяне на помощ и поддръжка за бързо отстраняване на бъгове в системата.

Добавяне на опция за печатане на договор/опис и справки.

Добавяне на отзиви и звездички , така клиентите ще могат да оценяват операторите и техните услуги.



Програмата поддържа система за известия. При вход на оператор излизат изкащи известия с информация за изтекли наемания. В раздел ‘Връщане на автомобил’ съответните наемания са оцветени в червен цвят. В долният ляв ъгъл на сцената се появява бледо-червен бутон, когато днес има изтичащи наемания. При натискане на бутона се появява нов прозорец с информация за дадените наемания. В таблицата те са оцветени в бледо-червен цвят.

За реализирането на маркирането в различен цвят е използван SetRowFactory.

rentTableView.setRowFactory(t -> new TableRow<Rent>() {

@Override

public void updateItem(Rent item, boolean empty) {

super.updateItem(item, empty) ;

if (item == null) {

setStyle("");

} else if (item.getDateReturn().isBefore(today)) {

setStyle("-fx-background-color: salmon;");

} else if (item.getDateReturn().isEqual(today)){

setStyle("-fx-background-color: #ffad99;");

}

}

});

**Клас AdminController** - Поддръжка на меню за админ.

Член функции

- addoperator() - Създава нов оператор

- addCompany() - Създава нова компания

**Клас carOpisController**

Член функции

- opisBtn() - Показва описи за избрана кола по регистрационен номер.

**Клас createCCController** - Добавя клас и категория

Член функции

- addClassification() - Добавяне на клас

- addCategory() - Добавяне на категория

**Клас loginController** - Управлява входа в системата

Член функции

- login() - вход в системата

**Клас OpratorMainView** - Поддръжка на меню за оператор

Член функции

- logout() - Изход от системата

**Клас RegistercarController** - Добавяне на автомобил

Член функции

- choosePicture() - Избира и добавя снимка от директория

- addNewCar() - Създава и добавя нова кола в системата

**Клас RegisterClientController** -

- addOperatorBtn() - Добавяне на нов клиент в системата

**Клас RentCarController** - Наемане на автомобил

- showClientInfo() - Визуализиране на клиенти в TableView

- initCartableView() - Визуализиране на коли в TableView

- imageViewUpdate() - Актуализира снимка

-rentBtn() - Съхранява данните за текущо наемане

**Клас ReturnCarController** - Връщане на автомобил

Член функции

- updateRentListView() - Актуализиране на данни за наемане

- calcRaiting() - Изчисляване на рейтинг на клиент

- calcPrice() - Изчисляване на дължима сума при наемане

- calcTravelKm - Изчисляване на изминати километри

- ReturncarBtn() - Съхранява данните за текущо наемане

**Тестове**

Тест за пресмятане на дължимата сума спрямо цена за изминати дни , клас на автомобила и възникнали проблеми.

С помоща на AssertEquals проверяваме дали функцията връща коректен резултат спрямо входните параметри.Ако параметрите не отговарят на зададените изисквания тестът се проваля.

Чрез Junit тестовете можем лесно да локализираме проблема и отстраняването му.

**public** **void** testCalcPrice() {

LocalDate dateRent = LocalDate.*of*(2019, Month.***DECEMBER***,1);

LocalDate dateReturn = LocalDate.*of*(2019, Month.***DECEMBER***,2);

Car rentedCar= **new** Car("","",**false**,**true**,0," ",**new** Classification("",1,1),**new** Category(" "));

Rent rent = **new** Rent(dateRent, dateReturn,**new** Client());

*assertEquals*(6.48,ReturnCarController.*calcPrice*(rentedCar, rent, 1,**false**),0);

}

Тест за коректност на функцията за пресмятане на рейтинг.

С помоща на теста потвърждаваме изчисленията на рейтинга.

@Test

**public** **void** testCalcRaiting() {

Client c = **new** Client("", "", "", 50, "");

*assertEquals*(30,ReturnCarController.*calcRaiting*(c, 20),0);

}

Аналогично на теста за рейтинга проверяваме изминатите километри.

@Test

**public** **void** testCalcTraveledKm() {

*assertEquals*(5, ReturnCarController.*calcTraveledKm*(10,5),0);

}

Тест за проверка дали клиентът съществува в базата данни , ако клиента е NULL тогава теста се проваля .

**public** **class** RegisterClientControllerTest {

@Test

**public** **void** testCheckCl() {

String pin = "9802026584";

Session session = rentacar.HibernateUtil.getSessionFactory().openSession();

session.beginTransaction();

Client c = (Client) session.createQuery("from Client s where s.clientPIN='" + pin + "'").uniqueResult();

session.getTransaction().commit();

assertFalse("Client is null",RegisterClientController.checkCl(c));

}

}