

НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

Департамент Информатика

CSCB734 Информационни системи "клиент - сървър"

Идеен проект за разработване на Web система

ТЕМА НА ПРОЕКТА:

**Система за управление на ремонти и поддръжка на
автомобили - Car Service**



F099841

Явор Станиславов Михайлов

Модул: Информатика

Специализация: Компютърно Програмиране

Програмен випуск: 2017 / 2018

Телефон: 0887 827 942

e-mail: yavor.st.m@gmail.com

1. Описание на системата Car Service

Системата има за цел да улесни, както собствениците на автомобили, да могат да записват часове в автосервизи и да проследяват сервизната история, така и на служителите в автосервизи, да проследяват, колите и клиентите с които са работили.

Потребителски роли и дейности

Чрез потребителските роли и дейностите, които те извършват, ще опиша какви са изискванията системата и как може да се взаимодейства с нея.

- Роля **Клиент** - потребител, който притежава автомобил/и и е клиент на автосервизи. Всеки клиент може да:
 - Се регистрира в системата
 - Добавя / Променя / Изтрива (CRUD) свои автомобили
 - Предварително записва час за посещения в автосервиз
 - Отменя запазени часове
 - Разглежда сервизна история на своите автомобили - предишни посещения в сервиз
 - Пише ревюта и да гласува с рейтинг за посетените автосервизи или за конкретно посещение
 - Сменя собствеността на автомобила - всеки клиент може да прехвърли собствеността на друг клиент, в случай на продажба
 - Добавя други собственици на даден автомобил, в случай, че автомобила се кара от повече от един човек

- Роля **Служител на сервиз** - потребител, който работи в сервиз.
 - Има определена квалификация, като услуги, които може да извършва - (напр. Смяна на гуми, Ремонт на ходова част)
 - Може да види списък с чакащи записани часове от клиенти
 - Може да види списък с всички клиенти на сервиза и съответно търси по име на клиента
 - Може да редактира чакащите записани часове, като добавя допълнително услуги, променя status-a, добавя цена на извършените услуги
 - Може да види списък с всички коли обслужвани в сервиза, в който работи. Също така може да филтрира списъка по регистрационен номер, година на производство или марка на МПС-то.
 - Може да види услугите, които са извършени на дадено МПС при всяко от посещенията

- Роля **Администратор на сервиз** - потребител, който има всичките правомощия като **Служител на сервиз**, като към тях се добавят:
 - Може да добавя / променя / изтрива служители от сервиза и техните данни
 - Може да променя данните за сервиза, като Име, Адрес, Услуги, които се извършват

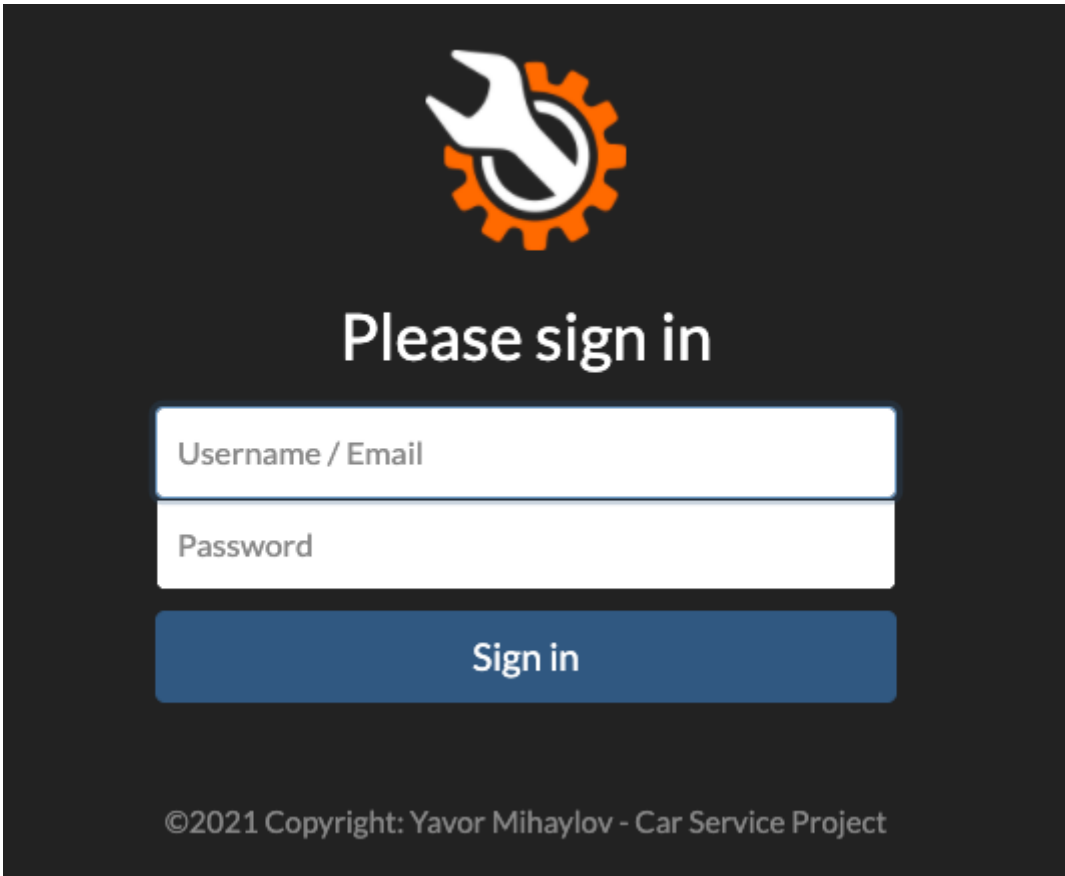
- Роля **Администратор на системата** - потребител, който има всички до сега изброени правомощия и достъп до

възможностите на системата, като той има достъп до всички сервиси в системата, може да разглежда данните им. Също така може да добавя нови сервиси и съответно Служители и Администратори.

Интерфейс на системата


Системата представя информацията чрез таблици и действията, които се извършват чрез бутони и хиперлинкове. За front-end е използвана библиотеката с предефинирани CSS стилове Bootstrap 4.0 - <https://getbootstrap.com/>. Следват няколко screenshot-а за да се добие по-ясна представа:

Login екран:




The login screen features a dark gray background. At the top center is a logo consisting of an orange gear with a white wrench and a white circle with a diagonal line through it. Below the logo, the text "Please sign in" is displayed in a large, white, sans-serif font. Underneath this text are two white input fields with blue borders. The first field is labeled "Username / Email" and the second is labeled "Password". Below these fields is a blue button with the text "Sign in" in white. At the bottom of the screen, the copyright notice "©2021 Copyright: Yavor Mihaylov - Car Service Project" is written in a small, white, sans-serif font.

СПИСЪК ОТ КОЛИ



Car Service Project

My Cars

Hello, client! 

My Cars

Add New Car

Registration Number	Make	Model	Year	Actions			
CB5029AM	Audi	A3	1998	Schedule Service Appointment	Edit	Delete	Show History
CB6985KC	VW	Passat	2010	Schedule Service Appointment	Edit	Delete	Show History
CB5029AM	Audi	Passat	1234	Schedule Service Appointment	Edit	Delete	Show History
CB5029AM	Alfa Romeo	passat b6	1234	Schedule Service Appointment	Edit	Delete	Show History

Форма за записване на час в сервиз

My Cars

Audi A3 CB5029AM -> Schedule Appointment

Repair Shop

Select Repair Shop

Date

dd.mm.yyyy

Time

--:--

Services

☐ Undercarriage

☐ Engines

☐ Transmissions

☐ Break systems

☐ Tires and Rims

☐ Oils and Filters

☐ Car Wash

☐ Wheel alignment

Save Changes

© 2021 Copyright: Yavor Mihaylov - Car Service Project

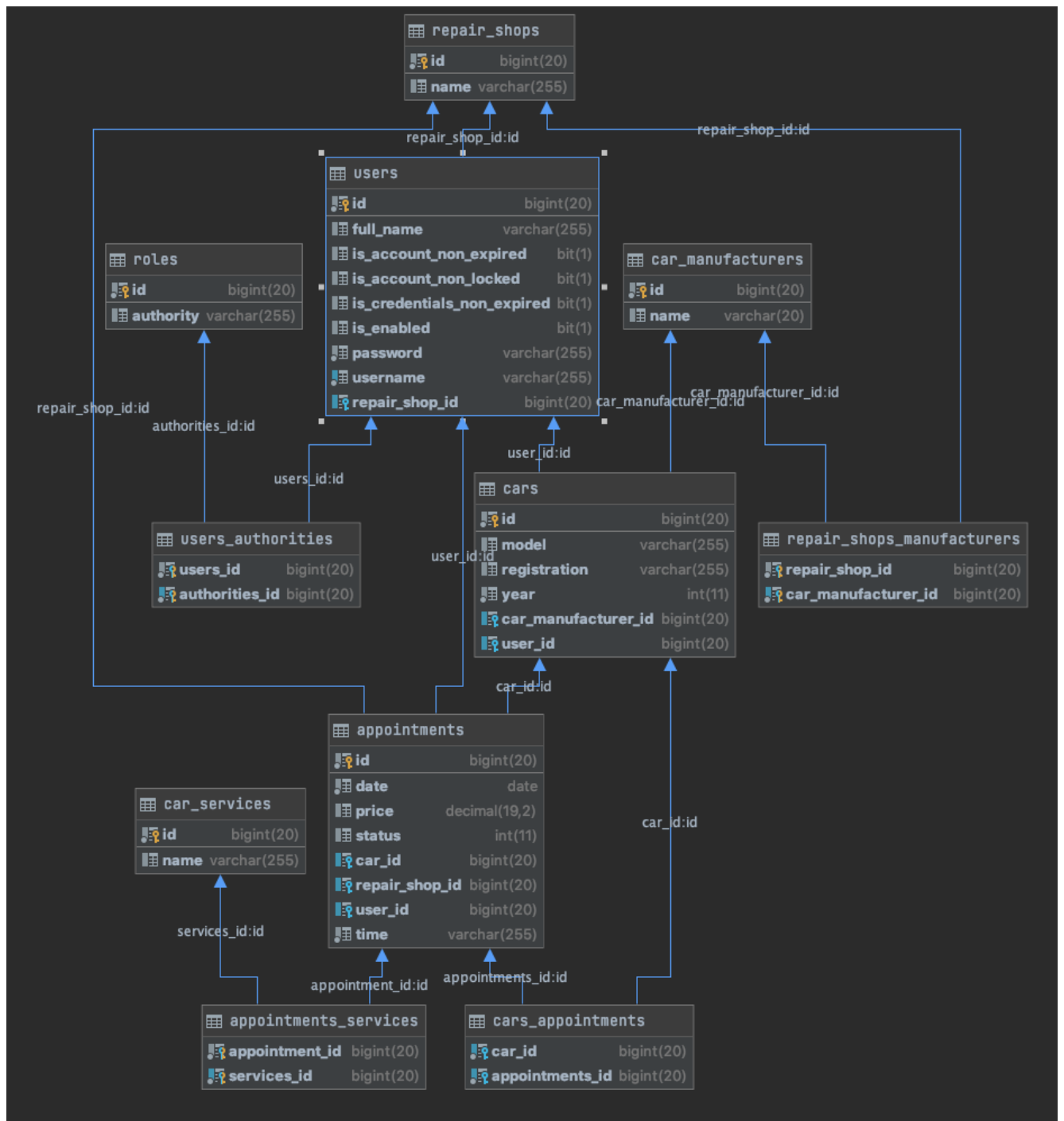
Сервизна история на даден автомобил

Audi A3 CB5029AM -> Appointments History						
< Back to My Cars List						
Date	Time	Repair Shop	Status	Price	Services	Actions
2023-03-12	12:10	Omega Service	PAID	250.00	<ul style="list-style-type: none">• Car Wash• Wheel alignment• Transmissions• Oils and Filters• Undercarriage	
2021-02-05	10:46	Omega Service	CANCELLED		<ul style="list-style-type: none">• Break systems• Tires and Rims• Engines	
2021-01-10	11:30	Omega Service	APPROVED		<ul style="list-style-type: none">• Wheel alignment• Transmissions• Oils and Filters• Undercarriage• Engines	

2. Схема на базата

Система работи с релационната база от данни MySQL. Главните таблици, с които системата работи са:

- **users** - съдържа потребителите на системата
- **cars** - колите в системата - има `user_id` за връзка към собственика на автомобила
- **repair_shops** - автосервизите в системата
- **car_services** - услугите, които се предоставят от сервизите и служителите
- **appointments** - съдържа историята на посещенията в сервиз
- **reviews** - реютата на клиенти за автосервизи или за дадено посещение
- **cars_co_owners** - други собственици на автомобили. Един автомобил винаги има един главен собственик. Може да има съсобственици. Тази таблица прави връзката с тях.



3. Избор на технология за реализация

За такъв тип Web-базирана система бяха избрани три технологии, с които може да се разработи. Следва да бъдат описани и да се изложат предимства и недостатъци на всеки от тях.

PHP - Laravel MVC Framework

PHP

Рекурсивен акроним от *PHP: Hypertext Preprocessor*. Open Source скриптов език. Използва се предимно за Web приложения, като 80% от публичните сайтове в интернет са написани на PHP. Въпреки, че се използва толкова масово за Web, в последната версия на езика се показва голям напредък в производителността при изчисление на сложни математически операции. Напредъкът се дължи на Just in Time (JIT) компилатора, който е въведен от версия PHP 8.0 на езика.

Laravel MVC Framework

Laravel е модерен MVC (Model-View-Controller) framework за web базирани PHP системи. Има изразителен и елегантен синтаксис, който е взаймстван от други модерни frameworks, като Ruby On Rails, Django (Python) и други. Направен е така, че да спести време на разработчика от скучните и повтарящи се неща във всеки проект за да може да се фокусира върху креативната работа свързана с бизнес логиката. Създаден е през 2011 година като оттогава до днес не спира да се развива. Има голям общност, като тя непрекъснато расте, поддържат се различни open-source пакети (модули) за различни цели и интеграции с framework-a.

Предимства

- PHP е доказан във времето си език за Web системи
- PHP и Laravel са лесни за учене
- PHP и Laravel имат добра и богата документация

- Има множество готови пакети (модули) - за Authentication, връзка със социални мрежи, изпълняване на миграции, обработка на csv/pdf и други
- Лесно се deploy-ва на всякаква среда (Portability) - Windows, Linux, Shared Hosting, Cloud

Недостатъци

- PHP е нетипизиран език, което от своя страна може да е предшественик на бъгове, ако разработчикът не внимава
- Биха имали performance проблеми, при много посещения и в такъв случай ще има нужда от допълнителен харудерен ресурс
- Въпреки, че документацията е добра, би отнело доста време програмиста да се запознае с нея, ако е нов със framework-а или езика
- Изискват солиден опит с технологията - трудно би било на един Junior програмист да започне веднага работа по системата
- Laravel търпи много промени от версия до версия и не е консистентен, дори и в minor update-ите си. Това би направило системата трудна за поддръжка, ако се иска framework-а да е постоянно с най-новата версия.
- В Laravel често възникват Security проблеми, които се докладват в мрежата

ReactJS (front-end) и NodeJS (back-end)

JavaScript

JavaScript, често се среща с абривиатурата JS, е скриптов език, който поддържа обектно-ориентиран и функционален стил

на програмиране. Създаден е от Netscape през далечната 1995 като от тогава до днес е претърпял много градивни промени. Стандартизиран е под името EcmaScript. JavaScript се изпълнява от браузъра (front-end) и се използва за да манипулира HTML (DOM дървото) и стиловете на HTML елементите - по този начин прави web страниците динамични. В последните години JavaScript се използва и за сървърни приложения, чрез технологията NodeJS.

Single-Page Applications

В последните години са модерни Single-Page Application-ите (SPA). Това са приложения, които взаимодействат със потребителя, като динамично се презаписва текущата web страницата със нови данни от web сървъра. При интеракция не се презарежда цялата страница, а само частта, в която данните се променят. При първоначалното зареждане на цялата страница, browser-а прави заявка към сървъра и в отговора на този request има налично цялото съдържание за показване на web приложението. Единствено след това се правят асинхронни заявки (AJAX) за различни данни. Целта на това нещо е по-бърз преход между страниците и създава впечатление за native приложение.

ReactJS

React е open-source, JavaScript библиотека за изграждане интерфейси за Web системи. Поддържа се от Facebook и от ReactJS общност - програмисти или цели компании, които я за използват. React може да се използва като база за single-page или мобилни приложения. Главната цел на библиотеката е

управление на състоянията на DOM дървото. За да се изгради функционално SPA приложение е нужно да се добавят и още библиотеки, като React Router (за url routing), React Redux и други.

NodeJS

Back-end JavaScript runtime среда, която е open-source и многоплатформена. Работи върху Chrome V8 Engine и изпълнява JavaScript код извън браузъра. V8 е engine-а, който изпълнява JavaScript в open-source Chromium браузъра на Google. Node представя парадигмата “JavaScript Everywhere” или “JavaScript навсякъде” - както в browser-а за динамични страници, така и за бизнес логика в back-end-а. Node.js е single-thread приложение, но може да поддържа конкурентно програмиране чрез концепцията си за events и callbacks (event-driver архитектура) - за това тази технология често намира място в изграждането на приложения с Web Sockets - Web чатове, Игри и други.

Предимства

- Използва се един и същ език за back-end и front-end - JavaScript
- JavaScript е лесен за научаване
- Основните неща се учат лесно и на опитен програмист би могло да му е лесно да навлезе в проекта
- React позволява лесно преизползване на цели Web компоненти
- Node е асинхронен процес, което позволява асинхронни I/O операции
- React поддържа виртуално DOM дърво, което от своя страна подобрява performance-а

- npm - JavaScript package manager - а. Има много готови пакети, които са open-source и са готови за използване

Недостатъци

- Технологията е много млада и не е сигурно дали след 5 години например, би се поддържала и използвала все още;
- JavaScript е нетипизиран език, което от своя страна може да е предшественик на бъргове, ако разработчикът не внимава;
- И React и Node не са лесни за учене в детайли. Изисква много обучение преди да се започне работа по същински проект;
- Няма богата документация, информация трябва да се търси от статии или уроци в интернет;
- React използва JSX. Това е tool, който миксира HTML и JavaScript. Това от някои програмисти се приема като предимство, но повечето го смятат като недостатък и лоша практика;
- Node не се скалира (Scalability);
- Трудно се работи с MySQL и въобще с релационни бази данни с Node;
- За да се разбере в дълбочина Node, трябва да се разбере добре как работи V8 engine-а;
- Node не е подходящ за интензивни процесорни задачи. Подходящ е само за тежки I/O операции (като web servers);

Java и Spring

Java

Java е обектно-ориентиран и типизиран език за програмиране. Създаден е с цел да се използва за различни цели (general-

purpose) - за Web приложения, системно програмиране, конкурентно програмиране. Java е designed така, че да работи върху всички платформи - Linux, Windows, Mac, Мобилни телефони, Embedded устройства и други. За това е нужно само устройството да поддържа Java (да има JVM) и приложението може да се стартира, без да е нужна компилация на ново. Според статистика през 2019 на GitHub, това е един от най-използваните езици за програмиране в световен мащаб за изграждане на Клиент-сървър web приложения, със преброени над 9 милиона разработчика. Java се поддържа от корпорацията Oracle.

Spring

Spring е open-source, Java базиран framework. С необходимите разширения и модули се използва за създаване на Web приложения и microservice-и. В него се спазват всички добри практики в разработването на софтуер и принципи като SOLID. За него има много готови разширения (extensions), които правят разработката на прости приложения лесни.

Предимства

- Java е обектно-ориентиран и типизиран език;
- Java се разработва от голяма корпорация (Oracle) и е доказана за времето си технология;
- Езикът Java по своята същност е максимално опростен и това го прави лесен за учене;
- Java и Spring са доказано сигурни;
- Java позволява конкурентно програмиране и е multi-threaded;
- Приложението ще е portable и може да се deploy-ва на различни среди. Java е platform-independent;

- Java е масов език - на всеки програмист първият език, който е учил в университет, училище или други курсове е бил Java. Шанса да се намерят програмисти, които да работят по системата е по-голям спрямо другите технологии;
- Java Spring Framework и допълнителните пакети и extension-ите спестяват много от работата рутинната и кода, който се пише при разработката и тестването на софтуера;
- Java е стандарт за Enterprise бизнес проекти;
- Тестването в Spring Framework е лесно;
- Конфигурирането в Spring Framework е лесно;
- Има добър built-in error handling в Spring;
- Spring има добър Dependancy Injection management;

Недостатъци

- Спрямо другите изброени технологии, Java е по-бавна (Poor Performance);
- Java има висока консумация на памет и сравнително по-бавна от natively компилираните езици като C или C++. Това за приложението не би било голям недостатък;
- Java е платена за комерсиален лиценз;
- Езикът е експресивен и изисква повече писане на код в дадени ситуация, които при други технологии биха отнели съществено по-малко редове код. Донякъде е предимство, донякъде недостатък;
- Spring Framework не е лесен за научаване в дълбочина, както и другите технологии;
- Spring дава много възможности на програмистите, но не трябва да се прекалява с използването им, защото това може да доведе до лош и неразбираем код;

- Spring се променя с времето и неща, които са били научени преди 2 години, вече може да не са релевантни;

Решение

Според изредените технологии и съвнение с предимства и недостатъци на всяка от тях, крайното решение е за изграждането този проект да се използва Java и Spring Framework. Предимствата, които изпъкват пред останалите са, че Java е масов език и се знае от много програмисти, поддържа се от голяма компания, която следи и за качеството на софтуера в цялата Java екосистема, Spring е сигурен и разработката на приложение става относително бързо.