|  |  |
| --- | --- |
|  | **Технически университет – София** |

**Курсов проект**

На тема:

**„Пректиране и разработка   
нa система за онлайн поръчване на храна за обяд от столова за хранене”**

Изпълнен от:

Десислав Андреев Христов

Факултет: ФКСТ

Специалност: КСИ

Факултетен № 121317006

Група: 222

Дата: 17.01.2018 г.

град София

Съдържание

[Глава. 1. Анализ на темата. 3](#_Toc504033984)

[1.1 Нуждата от подобно софтуерно решение. 3](#_Toc504033985)

[1.2 Описание на възможностите на реализираното приложението. 3](#_Toc504033986)

[Глава 2. Обзор на използваните технологии. 5](#_Toc504033987)

[Глава 3. Проектиране и архитектура на приложението. 5](#_Toc504033988)

[3.1 Схема на таблиците и релациите им в Базата данни. 5](#_Toc504033989)

[3.2 Софтуерна архитектута на приложението. 7](#_Toc504033990)

[3.2.1 Софтуерна архитектура Модел-Изглед-Контролер (Model-View-Controller) 7](#_Toc504033991)

[3.2.2 Комуникация между базата от данни и приложението 7](#_Toc504033992)

[3.2.3 Софтуерна архитектура Data Access Object (DAO). 8](#_Toc504033993)

[Глава 4. Основни функции и възможности на приложението и потребителски роли. 11](#_Toc504033994)

[Глава 5. Снимки на потребителския интерфейс на приложениетo 12](#_Toc504033995)

[Източници 13](#_Toc504033996)

[Линк към GitHub 13](#_Toc504033997)

# Глава. 1. Анализ на темата.

## Нуждата от подобно софтуерно решение.

В днешно време почти всички фирмени и държавни предприятния предлагат безплатен обяд на своите служители. Като най-честно те наемат някоя столова или ресторант, които могат да предложат вкусни и разнообразни ястия всеки ден. Това е добре за всички служители понеже те имат възможност да се хранят здравословно всеки ден, без да губят време да отидат до най-близкия ресторант (който може да бъде и не толкова близо). Много от хората обичат да си приготвят сами обяда ранно сутринта, но това отнема много време, а по този начин те могат отново да се хранят здравословно и без да губят време в приготвяне на обяд. Те просто отиват до столовата или ресторанта и си поръчват това, което желаят за обяд. А от друга страна столовата или ресторанта имат постоянна клиентела, която осугурява добър и постоянен доход на заведението. На пръв поглед изглежда, че всички са доволни, но всъщност не е точно така. Главната причина за това е защото всички хора обядват в едно и също време. Най-често това време е от 12:00 до 14:00. В тези два часа в заведението се натрупват големи опашки, а служителите на заведението се натоварват физически и психически до голяма степен с цел да обслужат клиентите по-бързо. И така обедната ти почивака се превръща в не чак толкова приятна, колкото ти се желае да бъде. Струпването на големи опашки е едно от най-неприятните неща за клиентите и служителите на заведенето. Друго неприятно нещо за клиентите е, че ако отидат да обядват по-късно(например след 13:30), много от ястията са вече свършили и те не могат да обядват това което желаят. Това се получава така, защото заведенето се опитва да предви колко бройки от дадено ястие е нужно да приготвят предварително и колко хранителни продукти да закупят за деня. Като по различни причини те почти никога не успяват да предвидят това. А също така понякога се получава и обратния вариант. Те приготвят твърде много от дадено ястие, и след това се налага да го изхвърлят. Причините за това са, че много от хората излизат в отпуска или пък работят от вкъщи, а те няма как да предвидят това. Естествено има много начини по които може да се избегне това. Като едно от решенията е системата „Lunchy“.

## Описание на възможностите на реализираното приложението.

Тази система е проектирана да бъде използвана както от клиентите на заведението, така и от служителите на заведението. Тя предоставя възможност за предварителни поръчки за обяд от клиентите, а служителите на заведението имат достъп до тези поръчки. Клиентите имат възможност да поръчат това което желаят за обяд от обедното меню за деня. Също така те могат да изберат час в който желаят това което са поръчали да бъде приготвено. По този начин те могат да бъдат сигурни, че когато отидат да обядват, те ще получат това което желаят, без да се притесняват, че може да е свършило. Понеже някой може да има доста работа в периода между 12:00 и 14:00, или просто да желаят да обядват по-късно. Това носи много плусове и за заведението защото по-този начин те могат да направят по-добри и точни изчисление за това колко хранителни продукти е нужно да закупят за деня, колко бройки от дадено ястие да приготвят и кога точно да започнат да ги пиготвят. Така те ще избегнат проблеми като приготвянето на твърде малко или трвърде много бройки от дадено ястие, също така ще намалят значително изхвърлянето на храна. Ще намалят значително напражението върху своите служители в обедния период от време(12:00 – 14:00). И най-вече ще намалят своите финансови разходи до минимум. А да не забравяме, че клиентите ще бъдат много доволни от бързото обслужване и от това, че няма да чакат на опашка за обяд и ще се насладят пълноценно на своята обедна почивка. Клиентите имат възможност да избират час за всяко едно от поръчаните ястия. Така един клиент например може да поръча супа, салата и основно ястие за 12:45 и десерт за 15:30. Задаването на час в който поръчката да бъде готова е по избор на клиента, тоест той може и да не избира конкретен час, а само ястия.

А за по-мързеливите клиенти или тези които нямат възможност да слязат за обяд има и опция за доставка на обяда до работното бюро. Като тази опция е възможна само ако даденото заведение предлага тази услуга. Също така една много добра и полезна функционалност на системата е, че клиентите имат възможност да следят в реално време статуса на всяко едно ястие от своята поръчка. Статуса който виждат клиентите когато първоначално направят своята поръчка е „Поръчката е приета“. След това когато вече се приготвя тя преминава в статус „Приготвя се“. И финалния статус е „Поръчката е готова“. Така дори да има някакво забавяне от страна на заведението, клиентите ще бъдат информирани и ще слязат за обяд когато е наистина готово. Когато клиентите разглеждат и избират какво желаят за обяд, те имат достъп до много подробна информация за различните ястия. Като например от какви хранителни продукти е приготвено ястието, описание и история на ястието, снимка и цена.

Служителите на заведението имат възможноста да добавят ново меню и нови ястия в системата, от които клиентите да избират по-късно. Те също така имат достъп до всички поръчки до момента. Така те имат нужната информация и могат да съобразят кога, в колко часа и колко ястия е нужно да приготвят. Понеже системата се използва само от избрани потребители и не е достъпна за публичен достъп, е нужно всички хора, които ще използват системата да бъдата първоначално добавени. За тази цел е нужно да има администратор на системата. Той има възможност да добавя нови потребители, като това могат да бъдат както клиенти така и служители на заведението.

# Глава 2. Обзор на използваните технологии.

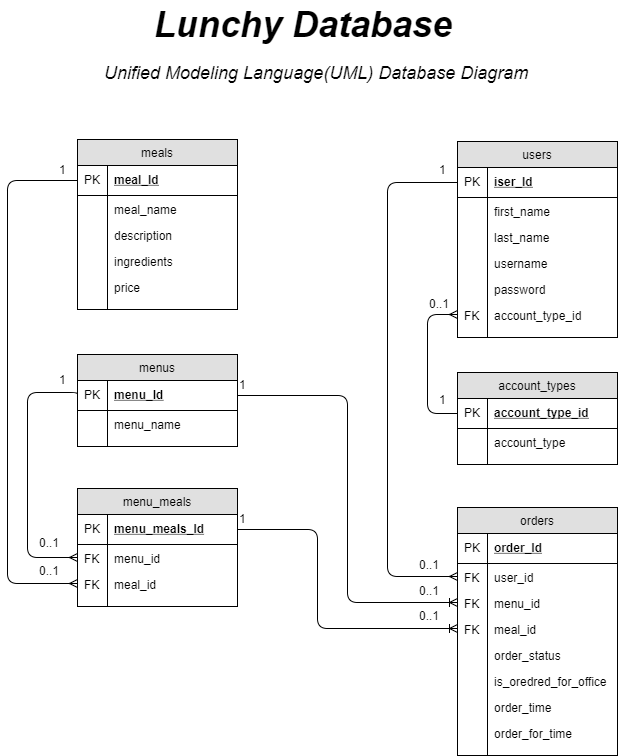
Технологиите които съм използвал за реализирането на тази система са:

1. Java EE
2. HTML
3. CSS
4. Javascript
5. MySql database
6. JDBC
7. Maven
8. Git

За реализирането на сървър часта на тази система съм използвал Java EE. И по-конкретно за обработване на HTTPS заявките съм изплозвал Java EE Servlets, а за изобразяване на динамично съдържание на потребителя съм използвал Java EE JSP страници. За база от данни използвам MqSql, а за осъществяването на връзката между проложението и базата от данни използвам JDBC. За представяне и оформяне на съдържанието в приложението съм използвал HTML, CSS и Javascript.

# Глава 3. Проектиране и архитектура на приложението.

## 3.1 Схема на таблиците и релациите им в Базата данни.



*Фигура 1. Схема на релациите между таблиците в базата от данни.*

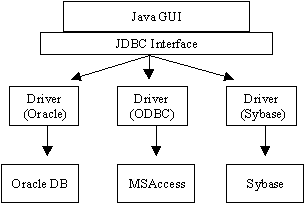
## 3.2 Софтуерна архитектута на приложението.

### 3.2.1 Софтуерна архитектура Модел-Изглед-Контролер (Model-View-Controller)

Системата е реализирана следвайки принципите на софтуерната архитектура Модел-Изглед-Контролер (Moldel-View-Controller). Това един от най-популярните и приложими в глобален мащаб шаблон за дизайн на уеб проложения. Той се основава на идеята за разделянето на бизнес логиката от графичния интерфейс и данните в дадено приложение. Той се състой от три компонента модел, изглед и контролер. Като моделa е ядрото на приложението, предопределено от областта, за която се разработва. Обикновено това са данните от реалния свят, които сме моделирали и над които искаме да работим - да въвеждаме, променяме, визуализираме и т.н.Трябва да се направи разлика между реалния обкръжаващ ни свят и въображаемия абстрактен моделен свят, който е продукт на нашият разум, който ние възприемаме във вид на твърдения, формули, математическа символика, схеми и други помощни средства. В тази система за моделите представляват класовете, които са реализирани следвайки програмния шаблон Data Access Object (DAO). Изглед е тази тази част от изходния код на приложението, отговорна за показването на данните от модела. В тази системата „изглед“ представляват JSP страниците. И контролера е тази част от сорс кода (клас или библиотека), която взима данните от модела или извиква допълнителни методи върху модела, предварително обработва данните, и чак след това ги дава на изгледа. Например може да бъде създаден един малък обект, в който да бъдат сложени данните за транзакцията - като в контролера бъдат взети данните за транзакцията от модела, бъдат преведени датите от UNIX формат във четим от потребителя формат, бъде преобразувана валутата от долари в евро например, бъде закръглено до втория знак вместо да се виждат данните както са в модела (и в базата) до 10-тия. Също така когато се прави уеб графичен интерфейс това би довело до много лесна модификация на HTML кода дори от човек, който не е програмист - той ще гледа на шаблона просто като на обикновена HTML страница. В тази система контролерите представляват Java класве наречени още сървлети(servlets). Те се грижат за това да модифицират и обработват данните, които да се представят на клиента, а също така да задействат добавянето на нови записи в базата данни.

### 3.2.2 Комуникация между базата от данни и приложението

Връзката между уеб приложението и базата от данни се осъществява благодарение на JDBC драйвер. JDBC (Java Database Connectivity) на Java Enterprise е първият междуплатформен програмен интерфейс за използване на бази от данни. Може да се каже, че JDBC е първият стандартизиран метод за интегриране на Java с базите от данни. В проектирането на JDBC са използвани основни абстракции и методи от ODBC. Идеята за базиране на JDBC върху ODBC идва от това, че ODBC е популярен сред независимите разпространители на софтуер и реализирането и използването на JDBC ще бъде по-лесно за хора, работили с ODBC. Най-общо казано архитектурата на едно приложение, използващо JDBC може да бъде видяна на фигурата, по-долу:



*Фигура 2. Архитектура на приложение, използващо JDBC.*

На фигурата може да забележите, че благодарение на тази технология, добавянето на нова база от данни, с която да работи приложението е много лесно. При проектирането на един софтуер е от основно значение системата да бъде лесно надграждана, чрез добавянето на нови функционалности. Подържането на друга база от данни също спада в тази категория, така че JDBC ни позволява тази гъвкавост.

### 3.2.3 Софтуерна архитектура Data Access Object (DAO).

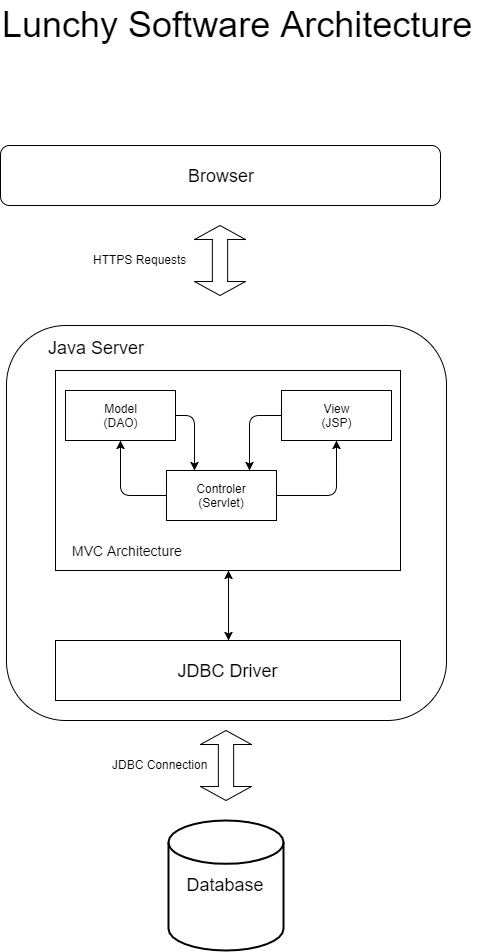
За да се реализира добър архитектурен дизайн за комуникацията с базата от данни, е препоръчително използването на шаблона Data Access Object (DAO). Това ще доведе до абстракция от начина, по който сървър приложението комуникира с базата и ще улесни добавянето на нова база от данни която да се поддържа от системата в бъдеще. Системата имплементира Data Access Object (DAO) шаблона за връзка с базата от данни. Този шаблон се използва за отделяне на начина на достъп до базата от данни (ниско ниво) от бизнес логиката на програмата (високо ниво). Този шаблон се състои от три основни компонента.

* Интерфейси – те дефинират набор от стандартни операции, които да се изпълняват от конкретните класове.
* Конкретни класове – те имплементират методите от горния интерфейс. Този клас е отговорен за вземането на данните от базата.
* Модели – Тези обекти са обикновени класове (POJO обект), които дефинират стандартни аксесори за достъп и модификация (getter и setter методи).

На следващата фигура може да разгледате примерна графична реализация на програмния шаблон Data Access Object (DAO). А на фигура 4 може да разгледате цялостната архитектура на системата „Lunchy“.



*Фигура 3. Примерна излюстрация на Data Access Object(DAO) софтуерна архитектура.*



*Фигура 4. Архитектура (структура) на системата.*

*.*

# Глава 4. Основни функции и възможности на приложението и потребителски роли.

Тази система притежава три типа потребителски акаунти и те са администратор, служител на столовата или ресторанта и клиент.

* Администратор

Администратора на системата притежава възможността да добавя нови потребители в системата, както клиенти така и служители на заведението. Той също така притежава достъп до менюто което предоставя заведението. Този тип акаунт е предвиден за ролята на управителя или собственика на заведението.

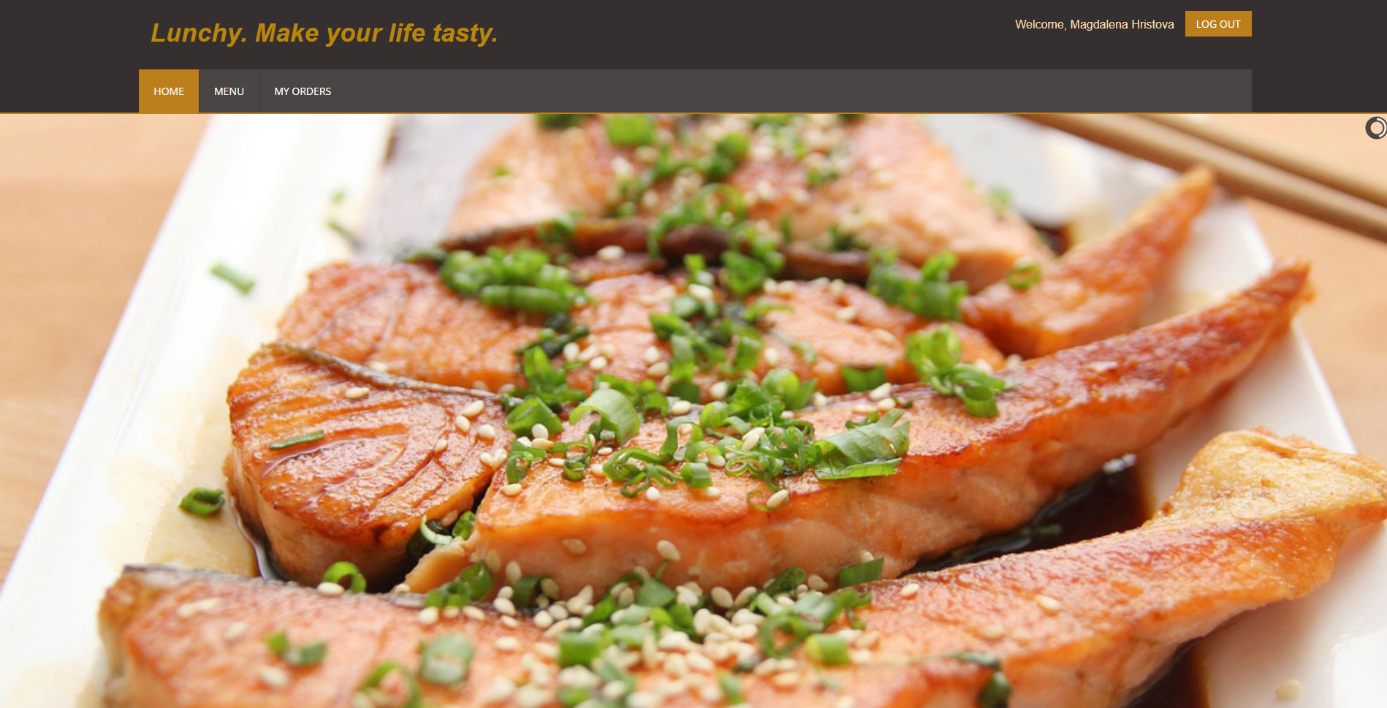
* служител на столовата или ресторанта

Следващия тип акаунт е за всички служители на заведението или поне за тези на които им е нужен достъп до всички поръчки. Това зависи строго от организацията на работа на персонала в самото заведение. Тези потребители имат достъп до менюто на заведението, до всички поръчки а също така имат възможността да добавят ново меню и нови ястия. В бъдещи версии на приложениете ще бъде добавена възможноста те да променят статуса на поръчките, но за момента се приема, че друга система ще променя статуса на поръчките. Основнато предназначение на този акаунт е, че той предоставя достъп до цялата информация относно заявениете поръчки в системата. Информация като номер на поръчката, клиента който е направил поръчката, от кое меню е поръчката, кои са поръчаните ястия, текущия статус, дата и час на поръчката, изберания час в който клиента желае да бъде приготвено и също така дали е поръчано за офиса.

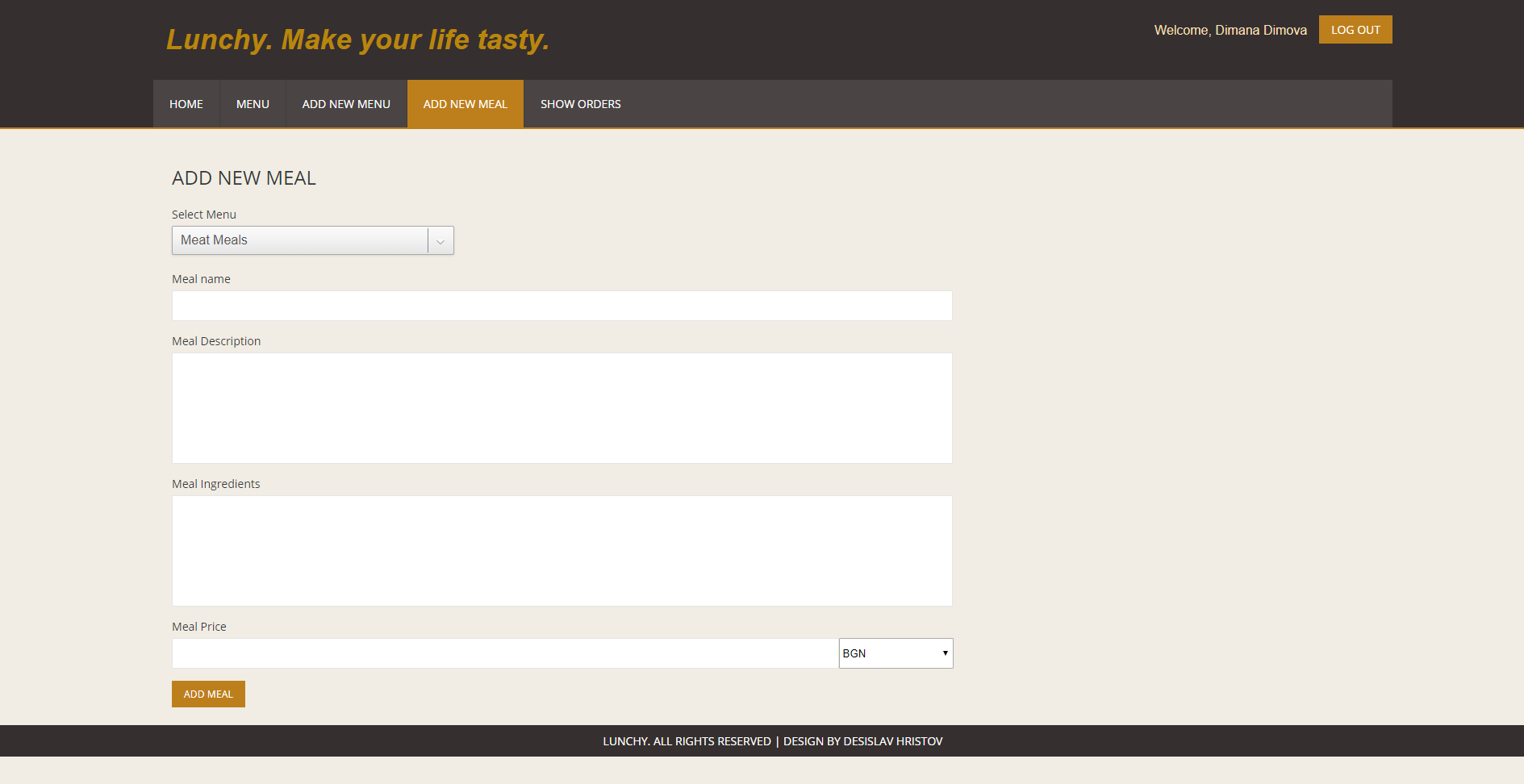
* Клиент на заведението

И последния тип акаунт в системата представляват всички клиенти на заведението. Те имат достъп до менюто на заведението, Имат полезна информация за всяко едно ястие, като например от какви съставки е приготвено ястието, описание и история на ястието, снимка и цена. Те имат възможност да си поръчат от всяко едно ястие предлагано в менюто, а също така могат да поръчат и за свой колега, като просто поръчат две бройки от дадено ястие. След като изберат ястията, които желаят за обяд, те трябва да отидат на страницата „моите поръчки“, от където имат възможност да изерат час в който желаят да бъде приготвено и дали желаят да бъде доставено до офиса. Също така ако са решили, че вече не искат дадено ястие или са го поръчали без да искат, то те могат да го изтрият от своята поръчка. След като са въвели своите предпочитания за всяко едно от поръчаните ястия е нужно да натиснат бутона „поръчай сега“ и заявката се изпраща и записва в системата от където служителите на заведението могат да я достъпят и да започнат работа по приготвянето и. Също така от тази страница „моите поръчки“ клиентите могат да следят статуса на своите поръчки и да видят общата сума пари на своята поръчка.

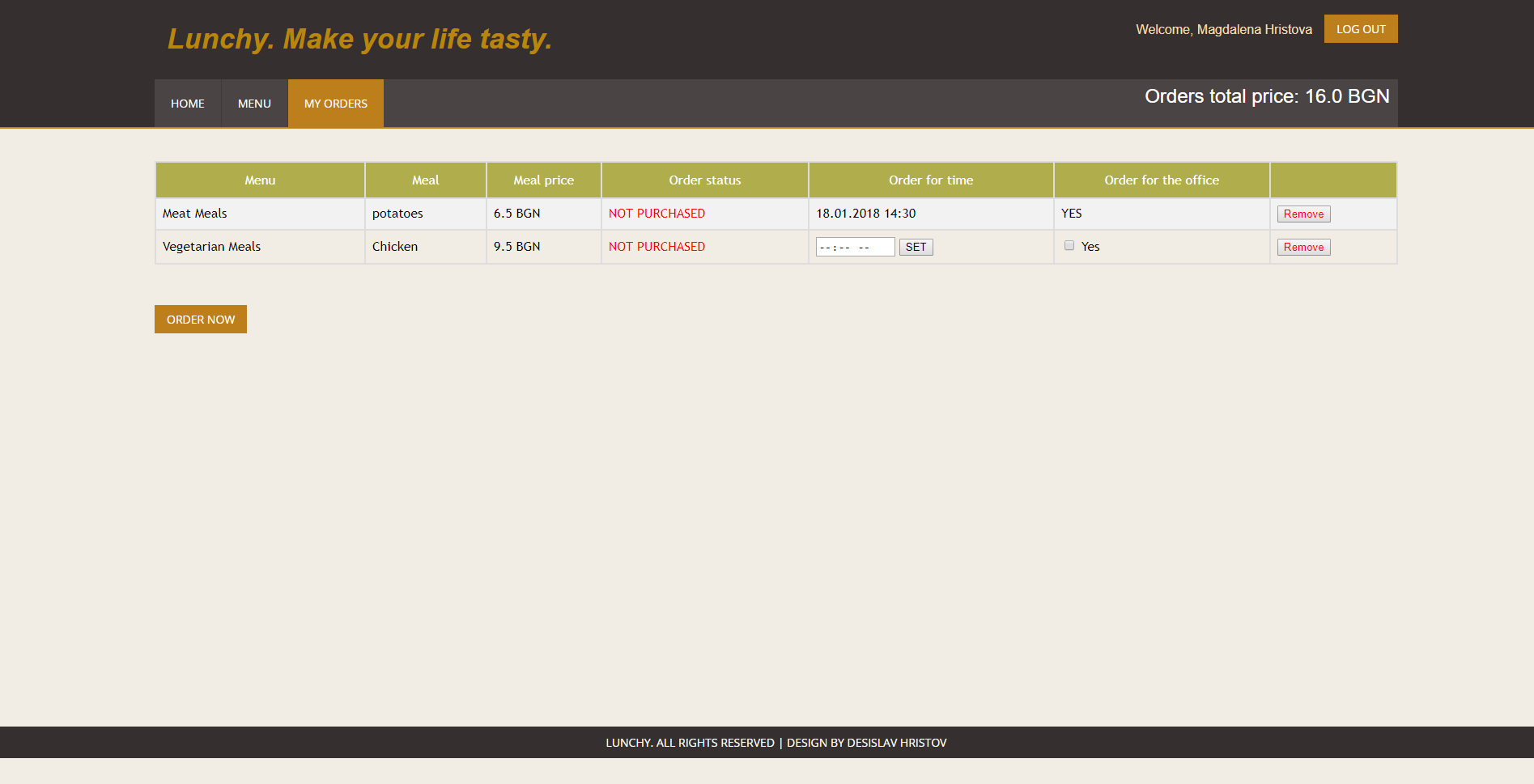
# Глава 5. Снимки на потребителския интерфейс на приложениетo



*Фигура 5. Начална страница на уеб приложението.*



*Фигура 6. Уеб страница за добавяне на ново ястие.*



*Фигура 6. Уеб страница за изпълняване на поръчка.*

# Източници

1. [Model–view–controller (MVC)](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller)
2. [Data Access Object (DAO)](https://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm%20)
3. [JAVA EE Servlets](https://docs.oracle.com/javaee/6/tutorial/doc/bnafe.html)
4. [JAVA EE JSP](https://www.journaldev.com/2021/jsp-example-tutorial-for-beginners)

# Линк към GitHub

* Линк към програмния код на приложението: [Lunchy code](https://github.com/desso919/lunchy)