



**ДЪРЖАВЕН ИЗПИТ  
ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА  
ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ – ЧАСТ ПО ТЕОРИЯ НА  
ПРОФЕСИЯТА**

**ДИПЛОМЕН ПРОЕКТ**

**Тема: „Сайт за създаване на резюме“**

**професия: 481020 „Системен програмист“  
специалност: 4810201 „Системно програмиране“**

**Дипломант: Владимир Петров Христатиєв, 12А клас  
Ръководител на дипломния проект: Милена Й. Добрикова**

**Подпис:**  
**(дипломант)**

**Подпис:**  
**(ръководител)**

**София, 2022 г.**

## С Ъ Д Ъ Р Ж А Н И Е

ВЪВЕДЕНИЕ .....	3
ОСНОВНА ЧАСТ.....	4
1. ИЗПОЛЗВАНИ ТЕХНОЛОГИИ.....	4
2. MVC МОДЕЛ И АРХИТЕКТУРА .....	11
3. MYSQL И СТРУКТУРА НА БАЗА ДАННИ.....	14
3.1 MYSQL И СТРУКТУРА НА БАЗА ДАННИ – РЕАЛИЗИРАНЕ НА CRUD ЗАЯВКИ .....	17
4. JSP КОМПОНЕНТИ И ВРЪЗКА С БАЗА ДАННИ.....	19
4.1. JSP КОМПОНЕНТИ И ВРЪЗКА С БАЗА ДАННИ – РЕГИСТРАЦИОНЕН КОМПОНЕНТ .....	20
4.2. JSP КОМПОНЕНТИ И ВРЪЗКА С БАЗА ДАННИ – ЛОГИН КОМПОНЕНТ .....	22
4.3. JSP КОМПОНЕНТИ И ВРЪЗКА С БАЗА ДАННИ – АДМИНИСТРАТОРСКИ ЛОГИН КОМПОНЕНТ И МЕНИДЖМЪНТ .....	23
4.4. JSP КОМПОНЕНТИ И ВРЪЗКА С БАЗА ДАННИ – ПОТРЕБИТЕЛСКИ СЕСИИ.....	25
5. JAVASCRIPT СТРУКТУРА.....	27
6. ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС.....	29
7. СТАРТИРАНЕ НА ПРОЕКТА .....	31
ЗАКЛЮЧЕНИЕ .....	34
ИЗПОЛЗВАНА ЛИТЕРАТУРА.....	35
ПРИЛОЖЕНИЯ .....	37

## **В Ъ В Е Д Е Н И Е**

В днешния непрекъснато променящ се пазар на труда, наличието на динамична автобиография с възможност за многократна употреба и актуализация е от решаващо значение за успешно професионално усъвършенстване.

CiVi ще бъде безплатно за използване решение на вековната дилема на създаването на смислена и добре структурирана автобиография. Уеб базиран, изграден с JSP, HTML, JS, CSS и MySQL инструментът ще е достъпен за всеки, който желае да създаде първото си CV, или за някой, който е добре запознат с индустрията и занаята.

Крайната цел на този инструмент ще е да предостави на потребителя изчистен и удобен интерфейс с цел създаването на автобиография, която след това да се използва посредством започване или продължаване на професионална кариера въз основа на нужди и способности.

Процесът на създаване на автобиография се очаква да е изключително бърз и лесен. Необходима ще бъде регистрация на профил, който ще позволи използването на инструмента и запазването на готово CV за бъдещи ревизии – възможно без нуждата от имейл верификация или абонамент.

Очакваните резултати са готова за използване (или принтиране) автобиография в формата на PDF файл, запазване на автобиографията за конкретния логнат потребител чрез MySQL, възможността за експортиране на текстовото съдържание на цялото CV и копиране на информацията в клипборд.

## О С Н О В Н А Ч А С Т

Този проект има за цел да покаже това, че уеб базиран конструктор на CV (автобиографии) може да бъде направен дори с вероятно остарели софтуерни решения като JSP (Jakarta Server Packages) и Tomcat Server, същевременно включвайки и модерни софтуерни решения като MySQL, HTML & CSS, JavaScript и Java. Демонстрира и подходящо решение на много важна и търсена услуга в нашия модерен и непрекъснато променящ се свят, а именно създаването на добре структурирана автобиография.

### 1. ИЗПОЛЗВАНИ ТЕХНОЛОГИИ

Причините, поради които бе избрана технологията реализираща локалния хостинг (localhost) и внедряването на Java сервлети (JSP), а именно Apache Tomcat са:

- Tomcat е с отворен код. Пуска се с публичен лиценз, което означава, че всеки може да използва основните файлове за разработване на лични или търговски приложения. Предлага се за безплатно изтегляне и инсталиране (open source).
- Tomcat се предлага с вградени опции за персонализиране. Разширяването на основното приложение предоставя на Tomcat повече предимство при разработване (flexibility).
- Tomcat осигурява допълнително ниво на сигурност на потребителя. Много компании биха искали да поставят своите данни зад firewall, който обикновено е достъпен от инсталацията на apache (security).

- Tomcat е съвместим с различни платформи, тоест може да се използва на операционни системи като Windows, Mac OS, Linux (cross-platform).
- Софтуерната фондация на Apache Tomcat предоставя редовни актуализации, предоставяйки корекции на грешки и бъгове, улеснявайки разработчици.

Apache Tomcat е най-широко разпространеният софтуер сред разработчиците на Java уеб приложения. Повече от 60% от Java приложенията използват Apache Tomcat. Когато става въпрос за сравняване на Apache Tomcat с други подобни контейнери за сървъри и сервлети използващи Java, има няколко конкурента, които могат да се разгледат.

Tomcat vs. Jetty:

Когато става въпрос за леки сървъри, Tomcat най-често се сравнява с Jetty. Jetty е HTTP сървър и контейнер за сервлети. Дълго време Jetty беше единственият инструмент, който може да работи във вграден режим. Към момента Tomcat вече може да работи и във вграден режим. И двата инструмента са с отворен код.

<b>Tomcat</b>	<b>Jetty</b>
Лицензиран с Apache 2.0	Apache + Eclipse Public License 1.0 лиценз
Висок пазарен дял (над 50%)	Пазарен дял от 8 до 12%.
Много добре документиран	Добре документиран
Възможност за вграждане (embeddable)	Възможност за вграждане (embeddable)

Фигура 1

Tomcat vs Weblogic:

Weblogic е търговски проект, разработен от Oracle Corporation и е лицензиран, изискващ закупуване на лиценз. Weblogic има няколко предимства пред Tomcat, като поддържа разпределени транзакции с помощта на мениджъри на транзакции, поддържа Enterprise Java Beans и предлага други функции, които го правят добър избор за търговски и корпоративни приложения. Ако едно приложение изисква функции на Java EE и има нужда от търговска поддръжка, Weblogic е правилният избор. Имайте предвид обаче, че не е безплатно!

<b>Tomcat</b>	<b>Weblogic</b>
Отворен код	Разработено от Oracle Corporation
Бърз старт и преразпределение	Поддържа Enterprise Java Beans (EJB)
Може да се използва свободно	Не може да се използва свободно (платен)

Фигура 2

JSP и сървлетите са предназначени само за прости уеб приложения, които могат да използват данни от формуляр и да правят връзки (в този случай връзка с база чрез MySQL).

JSP притежава много функционалност и поддържа всички видове Java обекти. Макар да има малък пазарен дял, производителността и мащабируемостта на JSP е добра, защото позволява вграждане на динамични елементи в HTML страница посредством и JavaScript и CSS имплементации.

Причините, поради които бе избрана технологията реализираща релационна база данни, базирана на structured query language (SQL), а именно MySQL, са:

- MySQL е съвместим с повечето операционни системи, включително Windows, Linux, NetWare, Novell, Solaris и други вариации на UNIX.
- MySQL предоставя възможност за стартиране на клиентите и сървъра на един и същ компютър или на различни компютри, чрез интернет или локална мрежа.
- MySQL е мащабируем и може да обработва повече от 50 милиона реда (rows). Това е достатъчно за обработка на почти всяко количество данни. Въпреки че ограничението за размер на файла по подразбиране е 4GB, може да бъде увеличено до 8TB.
- MySQL е много гъвкав, тъй като поддържа голям брой “embedded” приложения.
- MySQL е достъпен безплатно за изтегляне и използване от официалния сайт на MySQL.
- MySQL е сигурен и защитава чувствителни данни с криптиране на пароли.

Имайки това общо разбиране от различните избори от системи за релационни бази данни, MySQL е една от най-популярните и най-добре оптимизираните опции.

	Тип База Данни	Лиценз	Доку- ментация	Мащаб	Поддрж. струк- тура	Ниво на трудност
MySQL	SQL	GNU Public License	✓	Ver- tical	Струк- турирани	Лесна
Maria DB	SQL	GNU Public License	✓	Ver- tical	Струк- турирани	Лесна
Postgre	Obj.Orient. SQL	Отворен код	✓	Ver- tical	Струк- турирани	Трудна
Oracle	Multi- model SQL	Paid (платен)	✓	Ver- tical	Струк- турирани	Трудна
Mongo DB	NoSQL document orient.	SSPL лиценз	✓	Hori- zontal	Струк- турирани	Лесна

Фигура 3

Проектът е структуриран използвайки JSP (Jakarta Server Packages). JSP е вид сървлет на Java, който е проектиран да изпълнява ролята на потребителски интерфейс за уеб приложение на Java. JSP се изписва като текстов файл, които комбинира HTML или XHTML код, XML елементи и вградени JSP действия и команди. Използвайки JSP, може да се събере информация от потребителите чрез формуляри на уеб страници, да се представят записи от база данни или друг източник и да се създават уеб страници динамично.

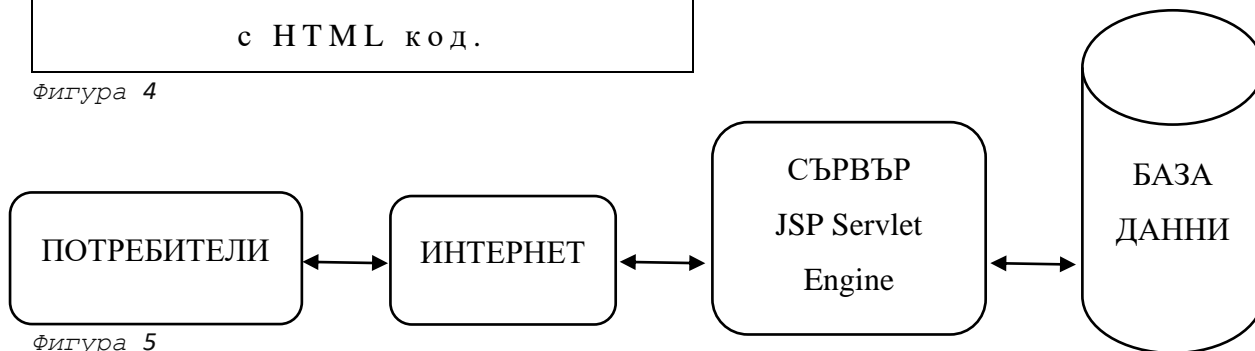
Като технология, JSP и сървлетите се считат, като остарели, с по-малък пазарен дял и употреба като цяло. Това не променя факта обаче, че те имат своето място в индустрията като силен конкурент, особено ако е нужно използването на Java, Jakarta EE (JEE) технологии т.е. JSP и сервлети.



Jakarta Server Pages страниците са изградени върху Java Servlets API и имат достъп до всички мощни Enterprise Java API инструменти, включително JDBC, JNDI, EJB, JAXP и др. Накратко, избрах да използвам тази технология за разработване на уеб страници, поради нейната гъвкавост, съвместимост и скорост. JSP може да съдържа всичко, което може да съдържа обикновена HTML страница, което го прави идеален за проект нуждаещ се от JavaScript, CSS и връзка с някакъв вид база данни.

Предимства на JSP	Недостатъци на JSP
Лесен за кодиране и поддръжка.	Проблемите са трудни за проследяване.
Висока производителност и мащабируемост.	Свързването с база данни не е лесно.
Лесен за поддръжка.	
Поддържа Java API.	
Лесно се използва и интегрира с HTML код.	

Фигура 4



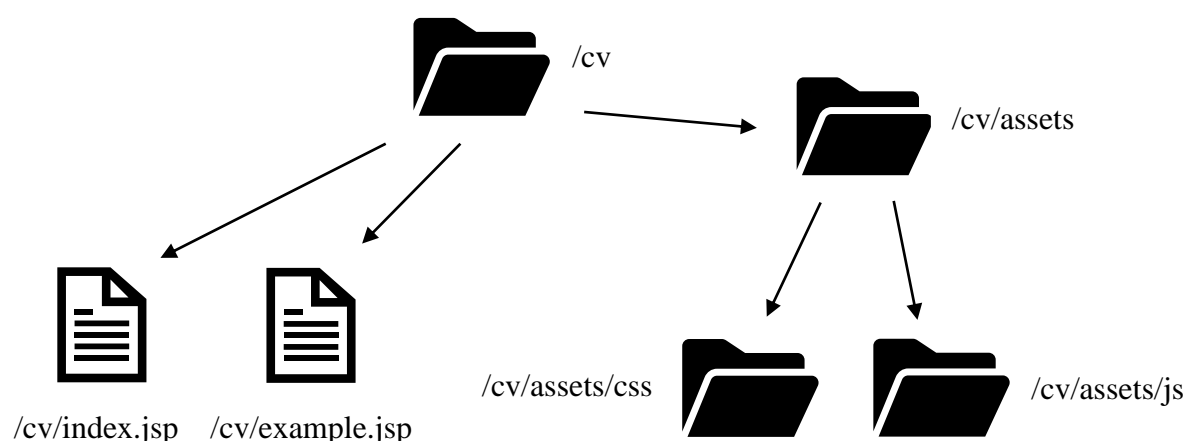
Фигура 5

\*Фиг.5 - Схематично обяснение на работата с JSP.

Посредством технологиите MySQL и JSP – финалният компонент нужен за изграждане на графичните и функционални аспекти на проекта е стандартният език на създаване на уеб страници - Hyper Text Markup Language или HTML.

HTML присъствието доминира в мрежата, като най-широко приетият език за уеб дизайн. Лесно се разпознава и интерпретира от популярните уеб браузъри (Chromium, IE 11, Firefox). Един от полезните аспекти на HTML е, че той може да вгражда програми, написани на скриптов език като JavaScript, който е отговорен за влиянието върху поведението и съдържанието на уеб страниците. Включването на CSS ще повлияе и на оформлението и външния вид на съдържанието на страницата.

Накратко, когато HTML са комбинира с JavaScript и CSS, е най-добрият възможен избор, когато става въпрос за проект с този обхват, цели и визия.



\*Фиг.6 - Схематичен изглед на проекта, съдържанията и разпределението на папките му.



\*Фиг.7 - Схематичен изглед на връзката между HTML, CSS и JavaScript и техните предназначения.

## **2. MVC МОДЕЛ И АРХИТЕКТУРА**

Модел-изглед-контролер (MVC) е архитектурен модел, който разделя приложението на три основни логически компонента: модел, изглед и контролер. Всеки от тези компоненти е създаден, за да обработва специфични аспекти на разработка на приложение. MVC представлява една от най-често използваните индустриални стандартни рамки за уеб разработка и създаване на мащабируеми и разширяеми проекти.

Компонентът Model съответства на цялата логика, свързана с данни, с които работи потребителят. Това може да представлява или данните, които се прехвърлят между компонентите View и Controller, или всякакви други данни, свързани с логиката.

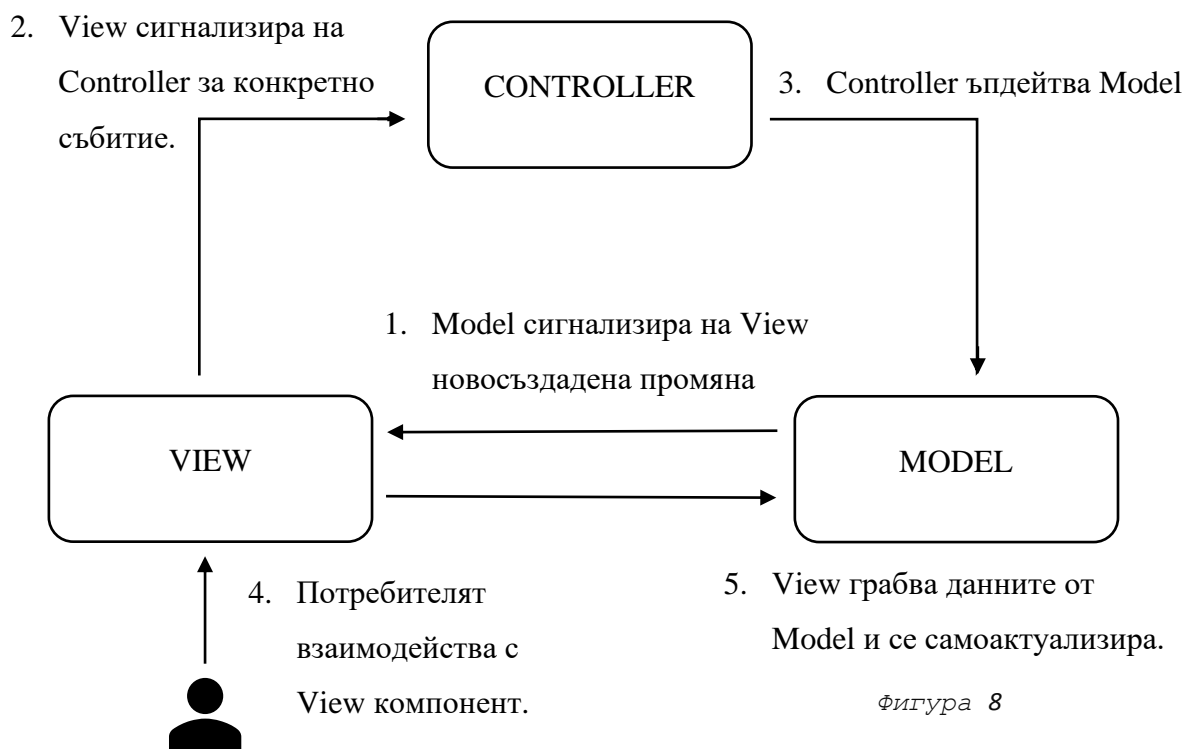
Компонентът View се използва за цялата логика на потребителския интерфейс на приложението. Например, изгледът на клиента ще включва всички компоненти на потребителския интерфейс, като текстови полета, падащи менюта и т.н., с които взаимодейства крайният потребител.

Контролерите действат като интерфейс между компонентите Model и View, за да обработват цялата бизнес логика и входящи заявки, да манипулират данни с помощта на компонента Model и да взаимодействат с View за изобразяване на крайния изход. Например, клиентският контролер ще обработва всички взаимодействия и входи от изгледа на клиента и ще актуализира базата данни, използвайки модела на клиента.

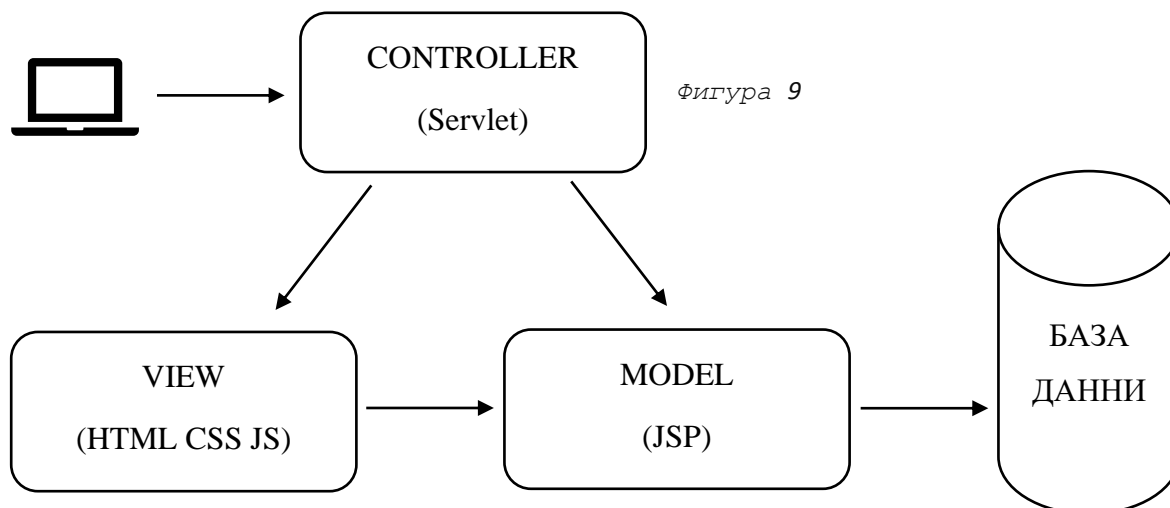
Когато става въпрос за използване на MVC модел, има няколко много важни предимства, които трябва да се вземат предвид:

- Организира уеб приложения с голям размер –  
Чрез MVC модел-а е възможно организирането и разделянето на кода на проекта, позволявайки лесни ревизии и лесно добавяне на нова функционалност.
- Лесна променяемост –  
Използването на MVC методологията позволява лесно модифициране на приложение. Добавянето/актуализирането е опростено в MVC модела (тъй като една секция е независима от другите секции).
- Лесно планиране и поддръжка –  
MVC модела е полезен по време на началната фаза на планиране на приложението, тъй като дава на разработчика схема как да подреди идеите си в действителен код. Помага и за ограничаване на дублирането на код и позволява лесна поддръжка на приложението.
- По-бърз процес на разработка –  
Чрез разделяне на кода на три нива, няколко разработчици могат да работят на отделни раздели от модела (Model, View или Controller).

Имайки предвид това, MVC архитектурата е подходящ подход при работа с уеб приложения, позволявайки правилно организиране и разделяне на кода въз основа на предназначението му.



\*Фиг.8 - Схематичен изглед на работата с MVC модел и начина, по който той функционира.



\*Фиг.9 - Схематичен изглед на работата с MVC модел в уеб базиран проект използващ JSP, HTML, CSS, JS и база данни.

### 3. MYSQL И СТРУКТУРА НА БАЗА ДАННИ

MySQL базата е структурирана на една схема (schema) и три таблици, като две от тях са логически свързани чрез foreign key. Трите таблици изграждащи схемата са:

- cv.login -

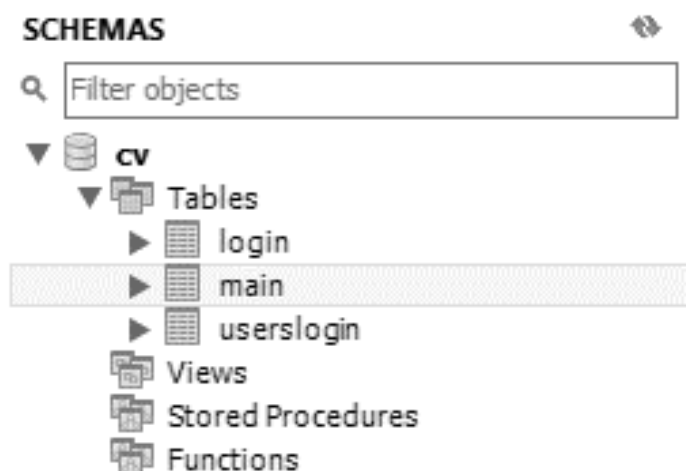
Таблицата login съдържа записи на администраторски профили и техните имена и пароли. Администраторските профили не могат да бъдат регистрирани, те се създават само и единствено от този, който е хост на услугата и съответно има достъп до MySQL базата чрез Workbench.

- cv.userslogin -

Таблицата userslogin съдържа записи на потребителски профили и техните имена и пароли. Потребителските профили съответно се регистрират от потребителите чрез регистрационната форма на уебсайта. Чрез мениджмънт страницата на сайта, администратор може да вижда и да трие записи в userslogin таблицата.

- cv.main -

Таблицата main пази потребителските записи от полетата за запис на информация при създаване на автобиография. “Email” идентификатора в userslogin и main е свързан чрез foreign key и се използва за различаване на потребителски профили и автобиографии. Също като userslogin таблицата, администратор може да вижда нейното съдържание и да трие записи чрез мениджмънт страницата на сайта.



\*Изглед на MySQL Workbench: Схема cv и таблиците cv.userslogin, cv.main и cv.login.

	WantedJobTitle	FirstName	LastName	Email	Phone
▶	test	test	test	test@test	123
	Frontend Dev	Vladimir	Hristakiev	vladimirhristakiev@gmail.com	0888201452

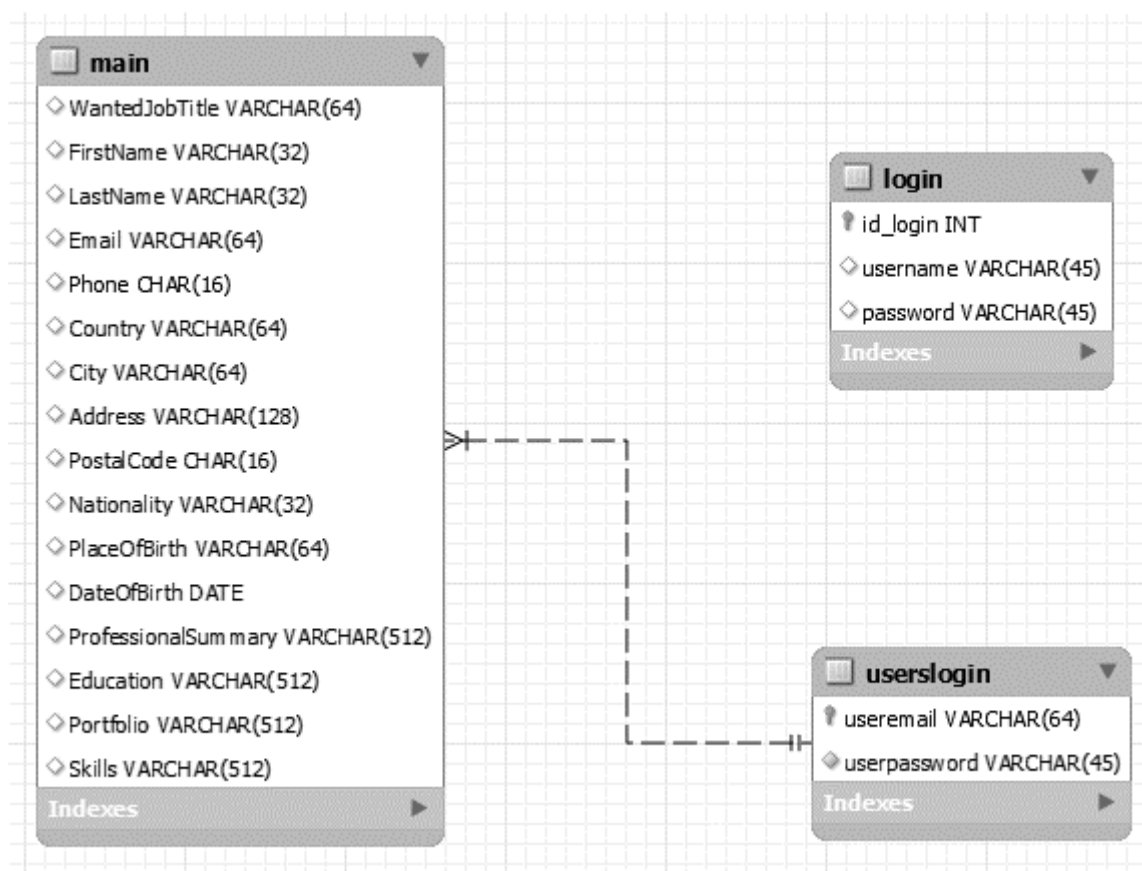
\*Част от изгледа на таблица cv.main с два съществуващи попълнени шаблони за автобиография.

	useremail	userpassword
▶	test@test	123
	vladimirhristakiev@gmail.com	123

\*Таблица cv.userslogin, съдържаща потребителски данни нужни за вход и използване на инструмента.

	id_login	username	password
▶	1	vlad	admin
•	NULL	NULL	NULL

\*Таблица cv.login, съдържаща администраторски данни нужни за вход при използване на мениджмънт страница.



Фигура 10

\*MySQL Workbench EER Диаграма на схема cv, таблиците cv.main, cv.userslogin, cv.login и foreign key връзка между таблица userslogin и main.

Структурата на базата е базирана на многофайлова релационна база данни (multi-file relational database). Релационна база данни съдържа множество таблици с данни с редове и колони, които се отнасят помежду си чрез специални ключови полета. Тези бази данни са по-гъвкави от плоските файлови структури и осигуряват функционалност за четене, създаване, актуализиране и изтриване на данни.



### 3.1 MYSQL И СТРУКТУРА НА БАЗА ДАННИ – РЕАЛИЗИРАНЕ НА CRUD ЗАЯВКИ

CRUD се отнася до акроним на обща парадигма, използвана при изграждането на уеб базирани приложения. CRUD означава: Създаване(Create), Четене(Read), Актуализиране(Update) и Изтриване>Delete). По точно, акронимът CRUD се отнася до съответните SQL функции като:

CRUD	SQL
Create	INSERT
Read	SELECT
Update	UPDATE
Delete	DELETE

Фигура 11

CREATE - INSERT заявка:

```
String Query = "INSERT IGNORE INTO main VALUES(";
java.util.Map<java.lang.String, java.lang.String[]> Fields = request.getParameterMap();
if (Fields != null)
{
    Integer Count = Fields.size();

    Integer Index = 1;

    for (java.util.Map.Entry<java.lang.String, java.lang.String[]>
        FieldEntry : Fields.entrySet())
    {
        String Value = (FieldEntry.getValue())[0];
        Query += "'" + Value + "'";
        if (Index++ != Count) { Query += ", "; }

        out.println(FieldEntry.getKey() + " - " + Value + "<hr>");
    }
    Query += ");";
}
else
{
    out.println("No map!");
}
```

\*INSERT заявка, генерирана от потребителските данни попълнени в шаблоните за създаване на автобиография.

READ - SELECT заявка:

```
String sql = "select * from cv.login where username=? and password=?";
PreparedStatement ps = con.prepareStatement(sql);
ps.setString(1, username);
ps.setString(2, password);
String unameDB="";
String passDB="";
ResultSet rs = ps.executeQuery();
```

\*SELECT заявка, извличаща информация от базата при сравняване на потребителски данни за влизане в профил.

UPDATE - UPDATE заявка:

```
String checkEmail = "UPDATE main SET Email='"+SessionEmail+"' WHERE Email IS NULL;";
```

\*UPDATE заявка, съществуваща като проверка за празно Email поле и попълване с потребителски мейл от сесия.

DELETE - DELETE заявка:

```
String Email = request.getParameter("d");
String no = Email;

Connection con;
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cv","root","admin");

Statement stat= con.createStatement();
stat.executeUpdate("delete from cv.main where Email='"+no+"'");
response.sendRedirect("management.jsp");

%>
```

\*DELETE заявка, използвана в администраторската мениджмънт страница, с цел изтриване на запазени потребителски шаблони за автобиография. Изтриването се извършва въз основа на идентификатор "Email".

## 4. JSP КОМПОНЕНТИ И ВРЪЗКА С БАЗА ДАННИ

Компонентите на проекта, изградени върху и изискващи JSP и връзка с база данни, са формуляри за регистрация и влизане (login ; register), проверки за потребителски сесии и страници за административно управление.

Всяка една от тези страници прави връзка с MySQL база данни чрез:

- Сесия (Session) - Имплицитен обект от тип HTTP Session -  
“HttpSession session1”
- Връзка (Connection) - “Connection x;”
- JDBC Driver -  
“DriverManager.getConnection(connector//localhost/schema,  
admin,pass);”
- SQL Заявка (Statement) – “String y = „\*Заявка\*”;“

```
HttpSession session1 = request.getSession(false);
Connection con;
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cv","root","admin");
System.out.println("Connected Login part");
String sql = "select * from cv.userslogin where useremail=? and userpassword=?";
```

Нужно е и правилно импортиране на java.sql технологии:

```
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.PreparedStatement"%>
```

Чрез JSP е възможно да се комбинират HTML и Java код, разделени от <% и %> оператори, позволявайки вкарването на HTML компоненти в if, for, while и т.н. с цел създаване на потребителски сесии, различни изгледи на страницата базирани на потребителски привилегии и различаване на потребителски профил от администраторски профил.

## 4.1. JSP КОМПОНЕНТИ И ВРЪЗКА С БАЗА ДАННИ – РЕГИСТРАЦИОНЕН КОМПОНЕНТ

CiVi използва регистрационна система за всички потребители на инструмента. За създаването на автобиография е нужно и създаването на нов потребителски профил, възможно чрез регистрационният компонент на уебсайта. Чрез новорегистриран потребителски профил (или вече съществуващ такъв), готови автобиографии се запазват в MySQL базата данни с опцията за бъдещи ревизии или корекции. Регистрационният компонент инкорпорира JSP код, който прави връзка с базата, като потребителският идентификатор в този случай е “email” вместо традиционното използване на прякори (nickname) или изискването на истински имена.

```
Connection con;  
Class.forName("com.mysql.jdbc.Driver");  
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cv","root","admin");  
System.out.println("Connected Login part");  
  
String sql = "insert into cv.userslogin(useremail,userpassword) values (?,?)";  
PreparedStatement ps = con.prepareStatement(sql);  
ps.setString(1, useremail);  
ps.setString(2, userpassword);  
  
ps.executeUpdate();
```

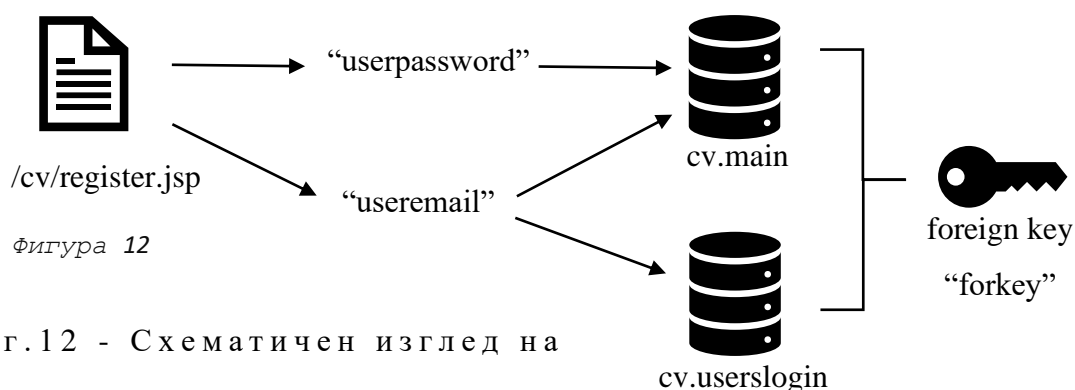
След осъществяването на връзка с базата чрез зададени име root и парола admin, заявката “sql” запазена като String, влага стойности в таблица cv.userslogin, като взима стойностите от потребителя в регистрационните полета на уебсайта.

```
<div class="row">  
  <div> <label>Email </label> <input type="email"  
    class="form-control" name="mail" required></div>  
  
  <div> <label>Password </label> <input type="password"  
    class="form-control" name="psw" required></div>  
  
</div>
```

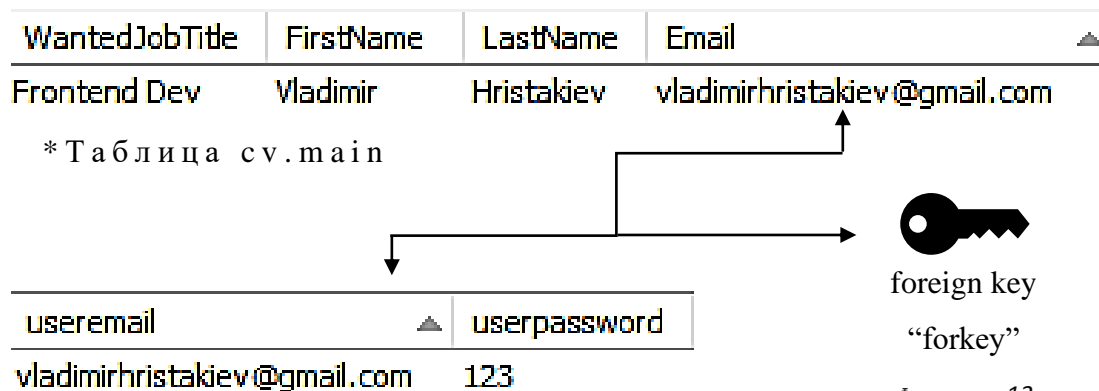
При създаване на нов потребителски профил чрез регистрационната компонента, с цел улесняване на потребителя, се създава вече готов празен шаблон в таблицата пазеща автобиографиите с попълнено “email” поле. Полето съответно се попълва от потребителската информация, приета от регистрационната форма в регистрационната компонента.

Това позволява новосъздаден потребителски профил да притежава правилно асоцииран шаблон за автобиография с вече попълнено “email” поле.

```
String regNewTable =  
"INSERT INTO cv.main VALUES('','','','"+useremail+"','','','','','','','"  
  
Statement.executeUpdate(regNewTable);
```



\* Фиг.12 - Схематичен изглед на връзката между регистрационната форма, таблица “cv.main”, таблица “cv.userslogin” и foreign key.



Фигура 13

## 4.2. JSP КОМПОНЕНТИ И ВРЪЗКА С БАЗА ДАННИ – ЛОГИН КОМПОНЕНТ

CiVi използва страница за вход (login) във връзка с компонента за регистрация. Страницата за вход позволява на потребителя да получи достъп до приложението (създаване на автобиография), като въведе своето потребителско име (в този случай “email”) и парола с цел удостоверяване. Формата на удостоверение (login authentication) изисква данните на вече съществуващ т.е регистриран потребител чрез регистрационния компонент. Подобно на страницата за регистрация, страницата за вход не изисква потвърждение по имейл, абонаменти или каквато и да е форма на проверка за легитимност (captcha). Логин страницата инкорпорира JSP код, който прави връзка с базата, като потребителският идентификатор в този случай е “email“. Идентификаторът работи в тандем с потребителската парола. При съвпадение на данните попълнени в логин формата от потребителя с тези запазени в MySQL базата, на потребителя е успешно предоставен достъп до създаването на нова или редактирането на стара автобиография.

```
HttpSession session1 = request.getSession(false);
Connection con;
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cv","root","admin");
System.out.println("Connected Login part");

String sql = "select * from cv.userslogin where useremail=? and userpassword=?";
PreparedStatement ps = con.prepareStatement(sql);
ps.setString(1, email);
ps.setString(2, password);
String emailDB="";
String passDB="";
ResultSet rs = ps.executeQuery();
```

\*Подобно на регистрационният компонент, създава се връзка чрез JDBC драйвер към MySQL база данни, с цел приемане на потребителски данни от логин уеб страницата.

### 4.3. JSP КОМПОНЕНТИ И ВРЪЗКА С БАЗА ДАННИ – АДМИНИСТРАТОРСКИ ЛОГИН КОМПОНЕНТ И МЕНИДЖМЪНТ

CiVi разполага с административна мениджмънт страница, която позволява директен контрол над cv.userslogin и cv.main MySQL таблиците. Като администратор, чрез мениджмънт страницата е възможно разглеждане и изтриване на записи в двете таблици. Достъп до тази страница изисква администраторски логин, като съществува само един администраторски профил изискващ отделна логин форма от тази на потребителите на сайта. Администраторски профил не може да се регистрира - може да бъде създаден единствено от самия администратор (т.е лицето, което има достъп до MySQL backend-а на проекта).

```
Connection con;
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cv","root","admin");
String sql="select * from cv.main order by FirstName";
Statement stat = null;
ResultSet res = null;
stat= con.createStatement();
res = stat.executeQuery(sql);
while(res.next()){
%>
<tr>
<td class="content_td"><%= res.getString("FirstName") %></td>
<td class="content_td"><%= res.getString("LastName") %></td>
<td class="content_td"><%= res.getString("Email") %></td>
<td class="content_td"><%= res.getString("Phone") %></td>
<td class="content_td"><%= res.getString("City") %></td>
<td class="content_td"><%= res.getString("Address") %></td>
<td class="content_td"><%= res.getString("PostalCode") %></td>
<td class="content_td"><%= res.getString("PlaceOfBirth") %></td>
<td class="content_td"><%= res.getString("DateOfBirth") %></td>
<td style="text-align: left; border-style: none; background-color:transparent">
<a href ='delete.jsp?d=<%=res.getString("Email")%>'
style="color:black;"><button>Delete</button></a>
</td>
```

\*Изобразяване на информация от MySQL таблици чрез HTML <td> или стандартна клетка за данни.

Административното изтриване на записи е възможно чрез два файла които изпълняват SQL заявки за триене на записи в редици, въз основа на конкретни аргументи.

```
String Email = request.getParameter("d");
String no = Email;

Connection con;
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cv","root","admin");

Statement stat= con.createStatement();
stat.executeUpdate("delete from cv.main where Email='"+no+"'");
response.sendRedirect("management.jsp");
```

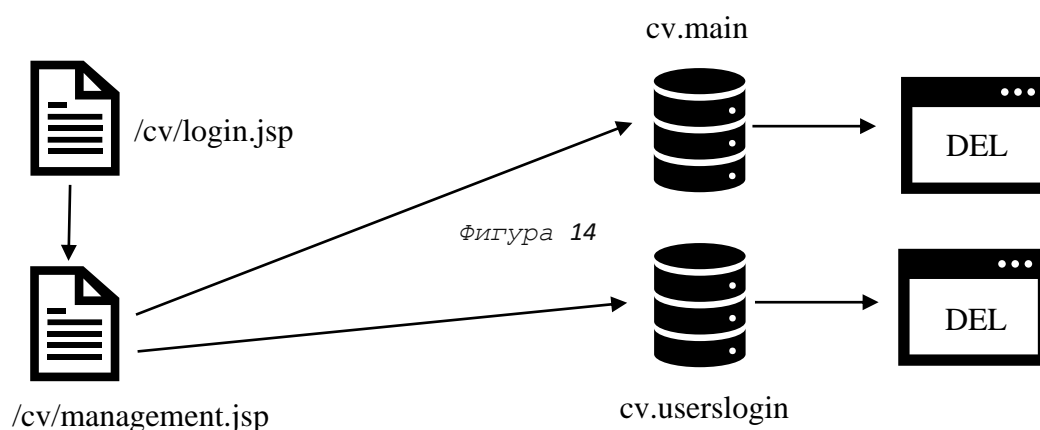
\*delete.jsp

```
String Email2 = request.getParameter("d");
String no2 = Email2;

Connection con2;
Class.forName("com.mysql.jdbc.Driver");
con2 = DriverManager.getConnection("jdbc:mysql://localhost:3306/cv","root","admin");

Statement stat2= con2.createStatement();
stat2.executeUpdate("delete from cv.userslogin where useremail='"+no2+"'");
response.sendRedirect("management.jsp");
```

\*deleteusers.jsp



\*Фиг.14 - Схематичен изглед на администраторски логин (login.jsp) към мениджмънт страницата (management.jsp), връзката с двете таблици и двете delete функции.



## 4.4. JSP КОМПОНЕНТИ И ВРЪЗКА С БАЗА ДАННИ – ПОТРЕБИТЕЛСКИ СЕСИИ

Потребителските сесии се използват за разграничаване между страница с влязъл (logged) потребител и страница без такъв. Целта на това е да се разграничат правата и изгледа на потребител, който не е влязъл, и потребител, който е. Създаване на сесия е възможно, чрез приемане на стойността на потребителски данни (в този случай името запазено под useremail), и сравняване на тази стойност в страниците, които изискват сесия за пълна функционалност.

```
con5 = DriverManager.getConnection("jdbc:mysql://localhost:3306/cv","root","admin");
String SessionEmail = (String) session5.getAttribute("useremail");
```

Приемането на потребителски данни за сесия се осъществява в невидим HTML footer, който се съдържа в отделен footer.jsp файл и се вика в всяка една страница индивидуално.

```
<%@ include file="footer.jsp" %>
    <%
        if(SessionEmail != null){
```

\*Вмъкване на footer.jsp в страница и проверка.

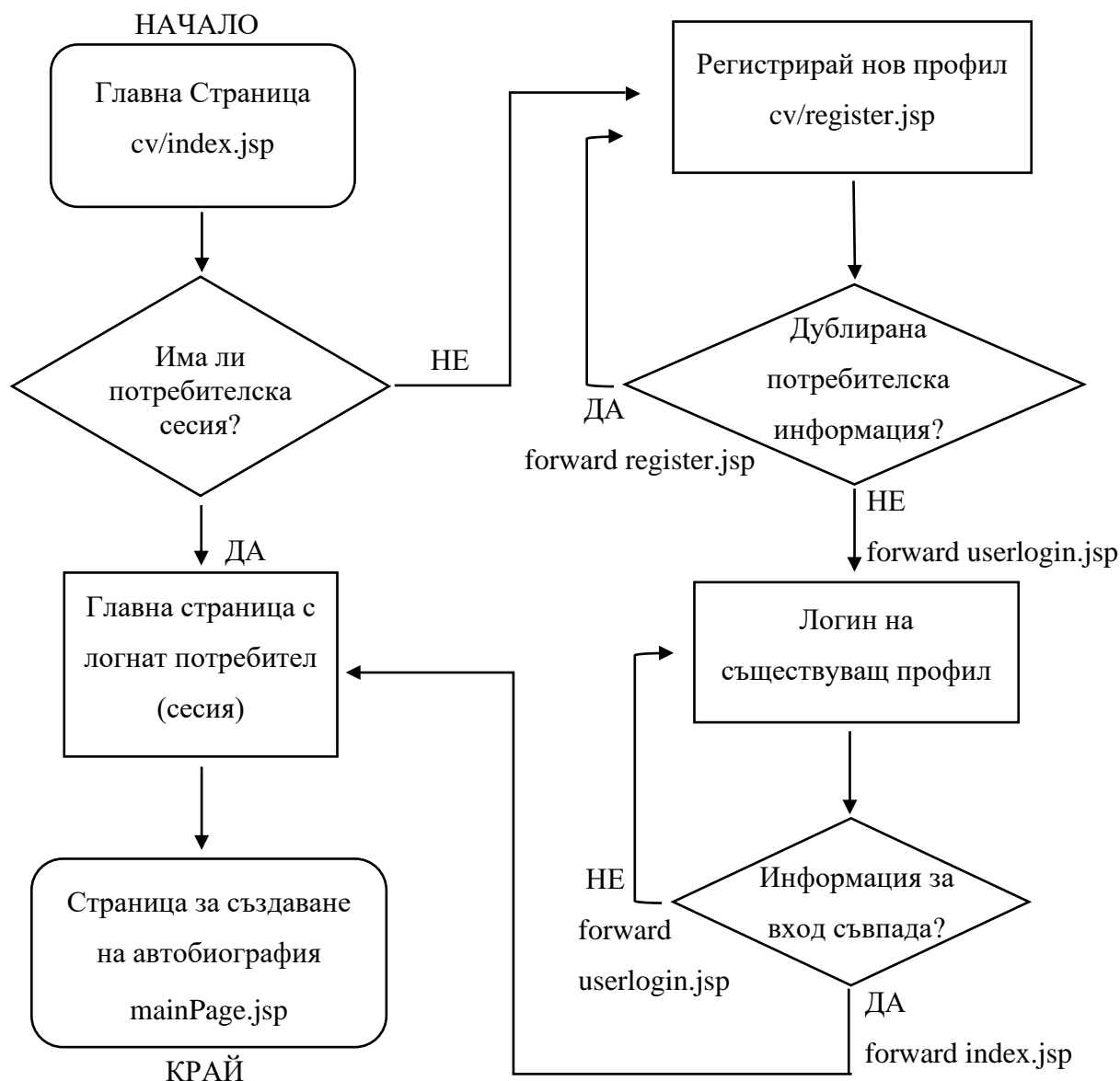
В случай на съществуваща сесия, потребителя получава достъп до опцията да излезе от профила си или да направи автобиография с избор от предварително изградени шаблони.

```
<%
if (session.getAttribute("useremail") == null || session.getAttribute("useremail").equals("")) {

%>

<div class="row">
<article class="col-4 col-12-mobile special zoom">
<a onclick="alert('Register or Login with an existing account to continue!')" target="_blank"
```

\*При празна променлива useremail или с стойност null, потребителя има ограничен достъп до страницата за създаване на автобиография.



Фигура 15

\*Фиг.15 - Схематичен изглед на процеса на проверка за потребителска сесия, създаване на профил, удостоверение на потребителски данни, проверки за повторения или грешки в потребителски данни и как е предоставен достъп към страницата за създаване на автобиография.

## 5. JAVASCRIPT СТРУКТУРА

JavaScript се използва за създаване на динамично и интерактивно уеб съдържание в уеб страници, като в този проект неговата роля е да предоставя функции нужни за елементи на потребителския интерфейс и логически функции свързани с потребителски действия.

Всички функции на JavaScript, използвани тук, са с отворен код и са съвместими с повечето съвременни браузъри (тествано на уеб браузъри, базирани на Chromium и Internet Explorer 11).

Функции свързани с интерфейс елементи позволяват анимирано скролване (scroll) при натискане на бутон от потребител.

За вмъкване и изобразяване на потребителска снимка върху страницата за създаване на автобиография:

```
<div id="display_image"></div>
```

\*Елемент съдържащ изображение.

```
const image_input = document.querySelector("#image_input");
var uploaded_image;

image_input.addEventListener('change', function() {
  const reader = new FileReader();
  reader.addEventListener('load', () => {
    uploaded_image = reader.result;
    document.querySelector("#display_image").style.backgroundImage = `url(${uploaded_image})`;
  });
  reader.readAsDataURL(this.files[0]);
});
```

\*JavaScript функция за приемане на потребителска снимка и изобразяване в контейнер “display\_image”.

За лесно копиране на изходната информация в случай на завършена автобиография от потребителя:

```
<a href="#" onclick="CopyToClipboard('sc');return false;">Copy Resume to Clipboard</a>
```

\*Хиперлинк референция към JavaScript функцията.

```
<script>
function CopyToClipboard(sc){

var r = document.createRange();
r.selectNode(document.getElementById(sc));
window.getSelection().removeAllRanges();
window.getSelection().addRange(r);
document.execCommand('copy');
window.getSelection().removeAllRanges();
}
</script>
```

\*Функция позволяваща копиране на данните на страницата, изключвайки HTML елементи като <hr> или <div> изписани в чист текст.

За предоставяне на потребителя възможността за отпечатване на автобиографията си чрез принтер или като .pdf файл.

```
<input type="button" id="button3" class=
"btnStyle no-print" onclick="window.print()" value="Print PDF"/>
```

\*Бутон който вика “window.print()” функция.

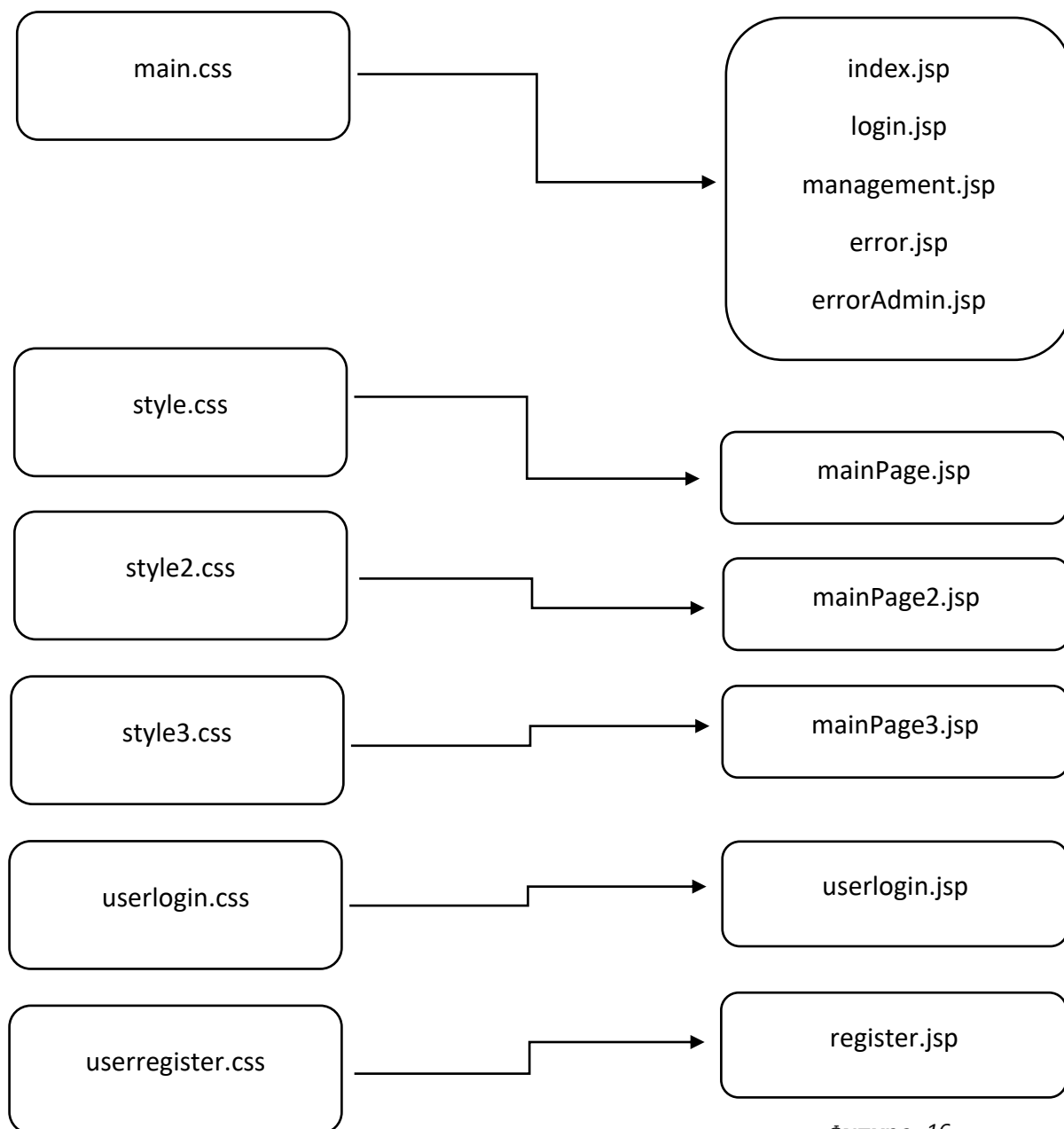
*Приложения – Фиг. П.6*

\*Функцията “window.print()” е вградена в JavaScript и се поддържа от всеки един модерен уеб браузър.

## **6. ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС**

Графичният потребителски интерфейс (GUI) е изграден използвайки HTML и CSS, като елемента на потребителско изживяване (UX) – чрез JavaScript елементи. Прилагат се няколко ключови принципа, често използвани когато става въпрос за дизайн на потребителския интерфейс:

- **Консистентни потребителски интерфейси -**  
Подобни модели на дизайн, идентична терминология, хомогенни менюта и екрани, последователни команди в интерфейса, консистентни цветове.
- **Ясност (Clarity) -**  
Интуитивна навигация между страниците, лесни за намиране и разграничаване бутони, лесно забележими интерактивни елементи, избягване на претрупани страници.
- **Принцип на обратна връзка -**  
Комуникация с потребителя и информирането му за действия, промени в състоянието на страницата, условия и грешки с кратък и прост език.
- **Гъвкавост (Flexibility) -**  
Оптимизация за различни устройства, резолюции и операционни системи/платформи.
- **Цветове и теория на цветовете -**  
Използване на привлекателни цветови комбинации, като се вземат предвид както психологическия аспект, така и естетиката.



Фигура 16

\* Фиг.16 - Схематичен изглед на графичните компоненти (css файловете) и техните връзки с JSP страниците.

CSS файловете и страниците асоциирани с тях използват “responsive” дизайн, позволяващ преглед с различни видове устройства и различни резолюции, избягвайки разваляне на съдържанието на страницата и нейните елементи.

```
/* For mobile phones: */
[class*="col-"] {
    width: 100%;
}

@media only screen and (min-width: 600px) {
    /* For tablets: */

}

@media only screen and (min-width: 768px) {
    /* For desktop: */

}
```

\*Пример на инкорпориране на “responsive” дизайн чрез CSS файл и “@media” аргумента за различни устройства.


## **7. СТАРТИРАНЕ НА ПРОЕКТА**

Стартирането (deployment) на проекта и имайки достъпа до всички негови функции е възможно чрез инструментите: MySQL Server, MySQL Workbench, Apache Tomcat Server и модерен Web Browser базиран на Chromium енджин или Internet Explorer 11.

Проекта използва MySQL ver. 8.0.27 и Apache Tomcat ver. 10.0.14, като двете стабилни версии са включени в основната папка на проекта. И двете услуги и техният сървър се стартират лесно чрез предварително конфигурирани “.cmd” файлове. В проекта са включени и начини на спиране на двете услуги (следователно спиране и на проекта):

*Приложения – Фиг. П.5*

Файлът “init.txt” съдържа SQL заявка, нужна за инициализиране на root потребител с специфични потребителски идентификационни данни, които ще оторизират (authorise) нов или съществуващ администратор, стартирайки MySQL Server-а и предоставяйки пълен контрол над базата данни.

 init.txt - Notepad

File Edit Format View Help

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'admin';
```

Файловете “mysql\_start.cmd” и “tomcat\_start.cmd” съответно стартират двете услуги, като викат техните вече съществуващи инсталации в root папката на проекта:

 \*mysql\_start.cmd - Notepad

File Edit Format View Help

```
@echo off
```

```
set PATH=mysql-8.0.27-winx64\bin;%PATH%
```

```
mysqld --console --user=root --init-file=  
F:\\java\\init.txt
```

```
pause
```

mysql\_start.cmd

 tomcat\_start.cmd - Notepad

File Edit Format View Help

```
@echo off
```

```
set JAVA_HOME=%CD%\JDK64
```

```
set PATH=%JAVA_HOME%\bin;%PATH%
```

```
set CATALINA_HOME=%CD%\apache-tomcat-10.0.14
```

```
%CATALINA_HOME%\bin\startup.bat
```

```
set CLASSPATH=%CATALINA_HOME%\common\lib\jsp-api.jar;%CLASSPATH%
```

tomcat\_start.cmd



Правилното стартиране (deployment) на проекта изрично изисква съществуването на стабилни версии на MySQL Server, Apache Tomcat Server и Java Development Kit (JDK). Липсата на тези изброени изисквания ще доведе до неадекватни или неочаквани резултати (невъзможност за стартиране на проекта, невъзможност за преглед на съдържанието на страницата поради ограничения от базата данни и т.н.)

Правилното следване на тези стъпки ще позволи достъп до основното уеб съдържание на проекта, достъпно чрез Apache Tomcat Localhost: <http://localhost:8080/cv/index.jsp>

*Приложения – Фиг. П.1*

## **ЗАКЛЮЧЕНИЕ**

Услуга като уеб конструктор за автобиография е нещо наистина ценно в наши дни. Демографската възраст, към която е насочен такъв инструмент, е доста широка и би послужила чудесно както за хора, които нямат опит, така и за тези, които имат.

По време на създаването на този проект бяха спазени множество лични цели, за да се постигне възможно най-доброто представяне и краен резултат:

- Правилно следване на CRUD методологията.
- Функционираща база данни с логически връзки.
- Правилно интегриране на JSP технология.
- Правилно следване на MVC модел на архитектура.
- Оптимизиран и бърз код, следващ конвенции за кодиране.
- Изчистен и опростен потребителски интерфейс.

Определено има много подобрения, които биха могли да са от полза за проекта - главно що се отнася до неговите графични елементи. Има много място за подобрение, когато става въпрос за шаблоните за дизайн на автобиография, оптимизации между различни устройства, резолюции и начина на изобразяване на страниците, по-добро имплементиране на потребителските сесии, повече административни функции и др.

## **ИЗПОЛЗВАНА ЛИТЕРАТУРА**

Интродукция на JSP -

<https://www.geeksforgeeks.org/introduction-to-jsp/>

23/04/2022

Предимства на JSP пред конкурентните технологии -

<https://flylib.com/books/en/1.94.1.95/1/>

06/04/2022

JSP - Общ преглед -

[https://www.tutorialspoint.com/jsp/jsp\\_overview.htm](https://www.tutorialspoint.com/jsp/jsp_overview.htm)

04/17/2022

MySQL Документация -

<https://dev.mysql.com/doc/>

23/04/2022

MySQL Документация Oracle -

[https://docs.oracle.com/cd/E17952\\_01/index.html](https://docs.oracle.com/cd/E17952_01/index.html)

16/04/2022

HTML Документация -

[https://www.tutorialspoint.com/html/html\\_tutorial.pdf](https://www.tutorialspoint.com/html/html_tutorial.pdf)

23/04/2022

Как да включва HTML в JSP? -

<https://stackoverflow.com/questions/16376227/how-to-include-html-in-jsp>

21/04/2022

Как да копирам в клипборда в JavaScript? -

<https://stackoverflow.com/questions/16376227/how-to-include-html-in-jsp>

23/04/2022

Window.print() в JavaScript -

<https://developer.mozilla.org/en-US/docs/Web/API/Window/print>

22/04/2022

Пример за JSP Servlet MVC с база данни-

<https://www.javaguides.net/2020/01/jsp-servlet-mvc-example-with-database.html>

19/04/2022

Визуален дизайн - теория на цветовете -

<https://dsource.in/sites/default/files/course/visual-design-colour-theory/downloads/file/visual-design-colour-theory.pdf>

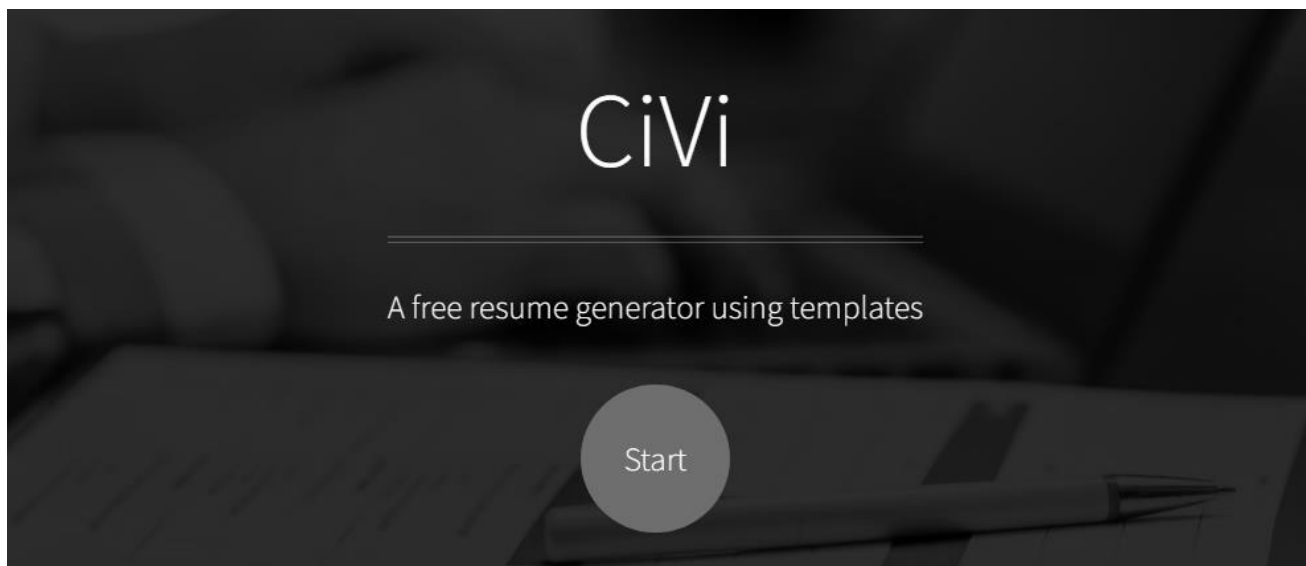
23/04/2022

Графичният потребителски интерфейс: Въведение -

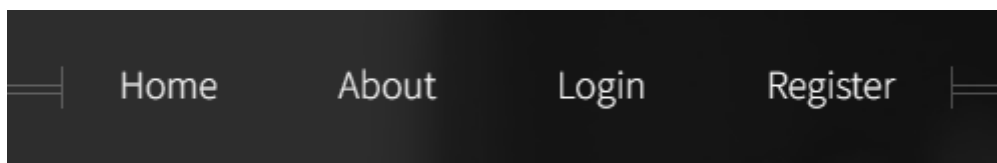
<https://faculty.ist.psu.edu/jjansen/academic/pubs/chi.html>

15/04/2022

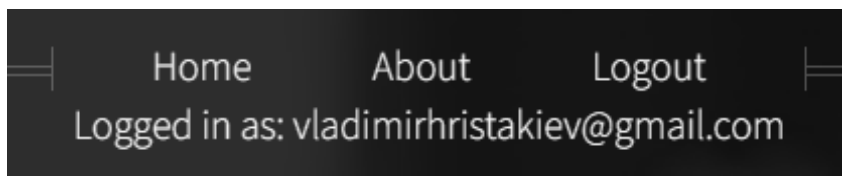
## П Р И Л О Ж Е Н И Я



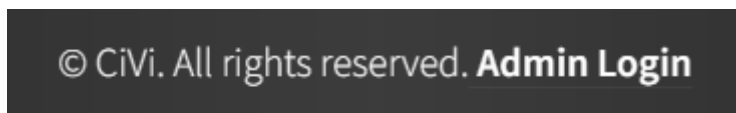
\*Фиг. П.1 Изглед на главна страница index.jsp














\*Фиг. П.2 Изглед на навигационен бар при сесия без потребител.



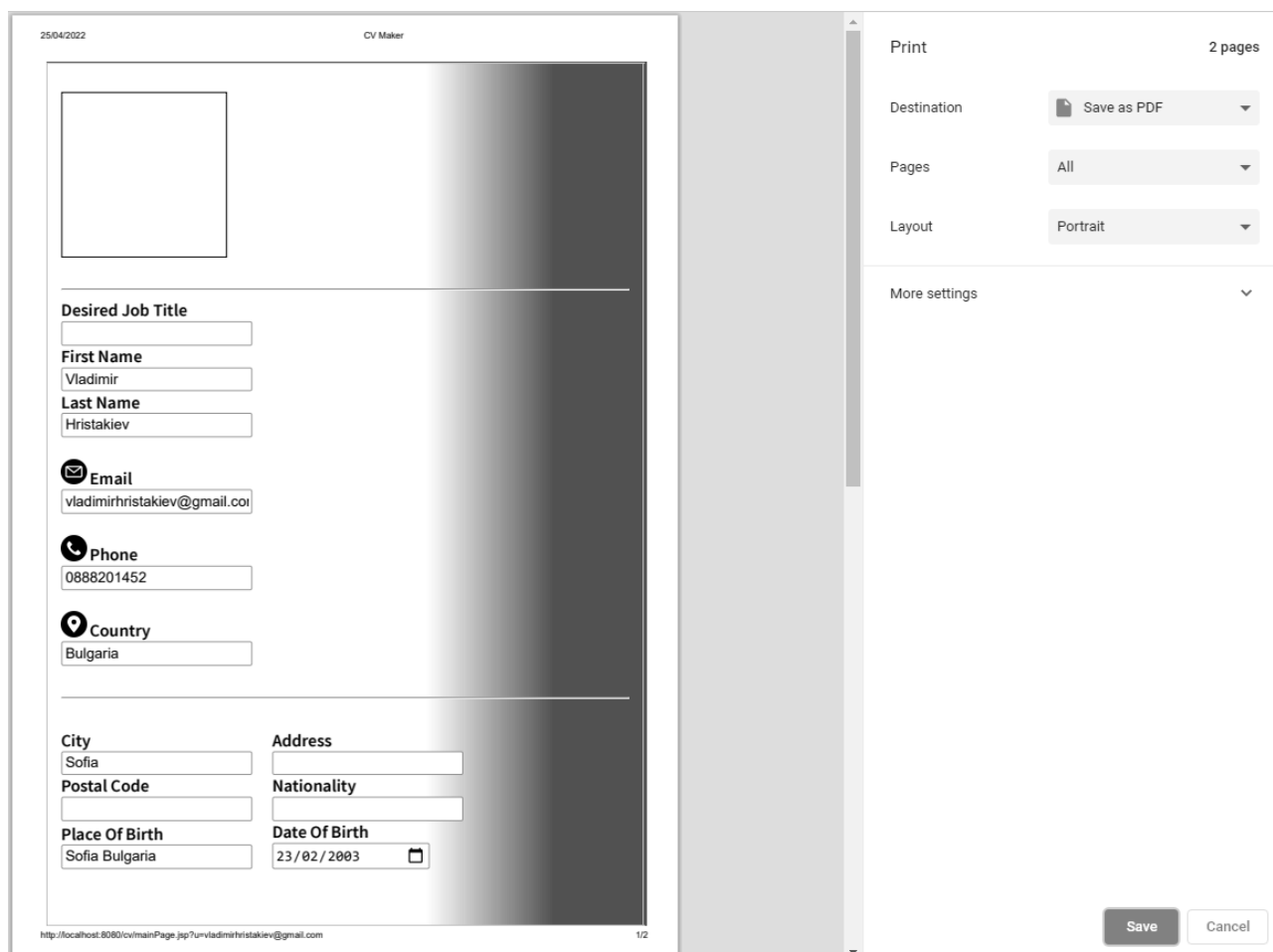
\*Фиг. П.3 Изглед на навигационен бар при сесия с потребител "vladimirhristakiev@gmail.com"



\*Фиг. П.4 Изглед на footer и бутон за администраторски вход.

Name	Date modified	Type	Size
 apache-tomcat-10.0.14	07/01/2022 12:53	File folder	
 JDK64	07/01/2022 12:43	File folder	
 mydata	11/01/2022 17:59	File folder	
 mysql-8.0.27-winx64	09/01/2022 02:56	File folder	
 CONSOLE	09/01/2022 21:16	Shortcut	2 KB
 init.txt	11/01/2022 17:56	Text Document	1 KB
 mysql_console.cmd	09/01/2022 21:16	Windows Comma...	1 KB
 mysql_start.cmd	30/03/2022 18:33	Windows Comma...	1 KB
 mysql_stop.cmd	10/01/2022 02:28	Windows Comma...	1 KB
 tomcat_start.cmd	10/01/2022 12:26	Windows Comma...	1 KB
 tomcat_stop.cmd	07/01/2022 13:10	Windows Comma...	1 KB

\*Фиг. П.5 Съдържание на главната папка на проекта. (root директория)



The screenshot displays a web application titled "CV Maker" with a date of "25/04/2022". The main form contains the following fields:

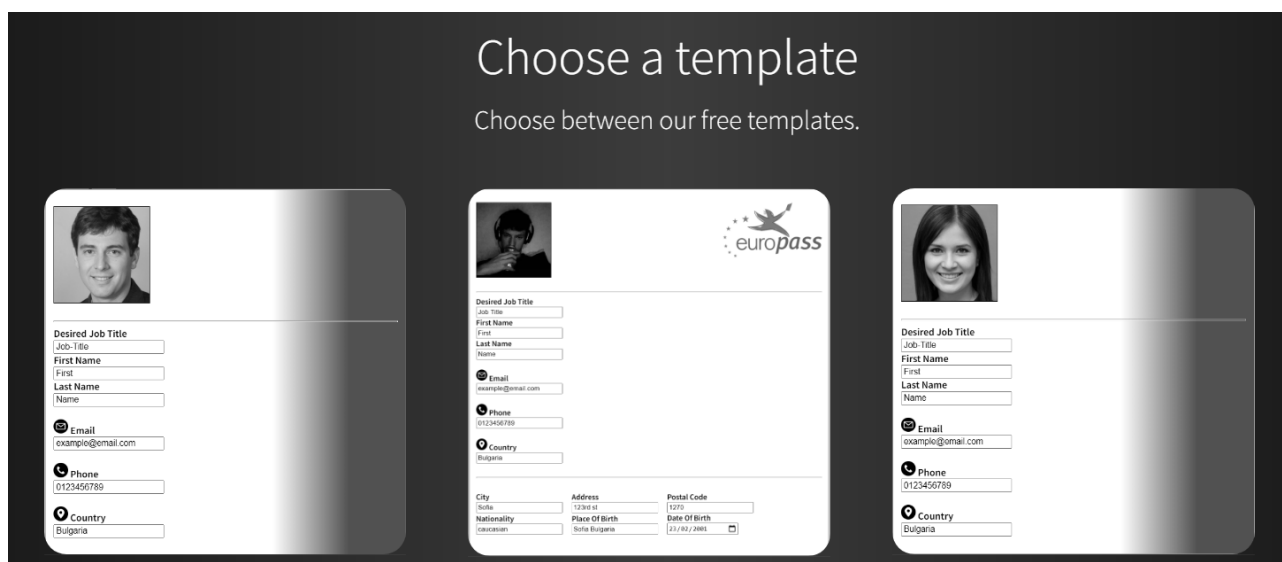
- Desired Job Title**: Empty text input.
- First Name**: Input with "Vladimir".
- Last Name**: Input with "Hristakiev".
- Email**: Input with "vladimirhristakiev@gmail.com".
- Phone**: Input with "0888201452".
- Country**: Input with "Bulgaria".
- City**: Input with "Sofia".
- Address**: Empty text input.
- Postal Code**: Empty text input.
- Nationality**: Empty text input.
- Place Of Birth**: Input with "Sofia Bulgaria".
- Date Of Birth**: Input with "23/02/2003" and a calendar icon.

On the right side, there is a "Print" sidebar with the following settings:

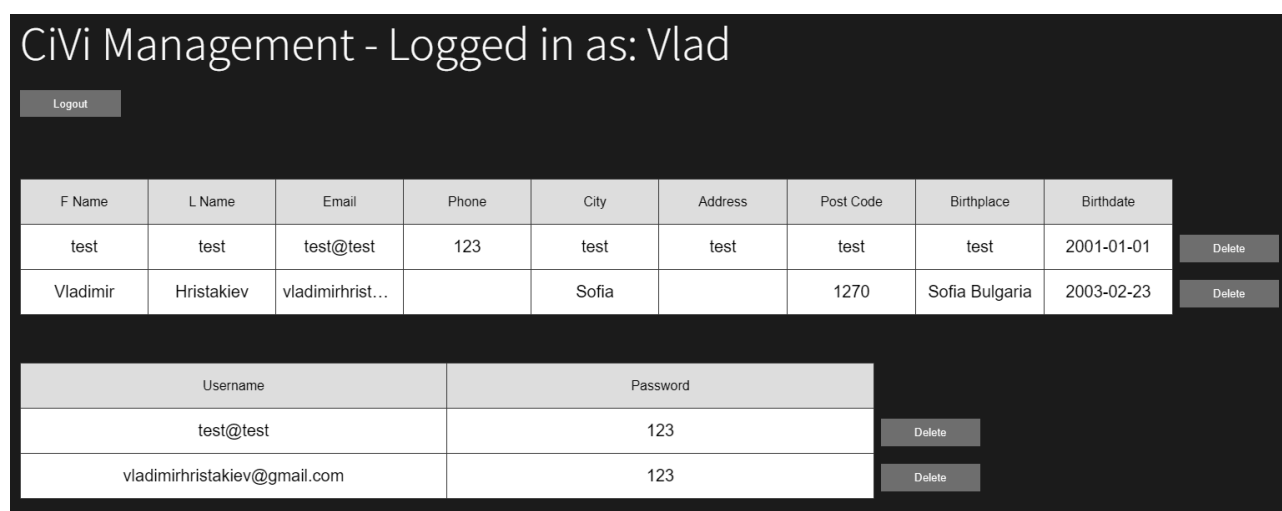
- Print**: 2 pages
- Destination**: Save as PDF
- Pages**: All
- Layout**: Portrait
- More settings**: Expandable menu.

At the bottom right, there are "Save" and "Cancel" buttons. The URL at the bottom left is "http://localhost:8080/cv/mainPage.jsp?uv=vladimirhristakiev@gmail.com".

\*Фиг. П.6 Изглед на window.print() функция в Chromium базирани уеб браузъри.



\*Фиг. П.7 Избор на шаблон за автобиография, с опция за стандартен Europass формат.



\*Фиг. П.8 Изглед на администраторска мениджмънт страница.