**Имена:**  Димитър Николаев Татарски **фн:** *81818***Начална година:** *2023* **Програма:** бакалавър, КН **Курс: 4**   
**Тема: 7.1 Filemeup  
Дата: 24.01.2023 Предмет: w19prj\_KN \_final имейл: tatarski.dim@gmail.com**

**преподавател:** доц. д-р Милен Петров

**Предаване:** Задачата се предава в архив с попълнен настоящия документ, проекта/проектите с кодовете, README.txt файл, който описва съдържанието на архива; папка с допълнителни компоненти и използваниресурси**, архива да се казва 9999\_project\_final.zip. (Успех!). (Редактирайте маркираните зони в жълто с коректната информация)**

# ТЕМА: 7.1 Filemeup

## 1. Условие

Система за качване и преглед на различни видове ресурси.

## 2. Въведение

Системата позволявам създаването на акаунт и качването в лична директория. Потребителя може да създава директории, да качва файлове и да разглежда вече записаните директории и файлове.

## 3. Теория

Системата работа на базата на архитектурата Model View Controller.

Моделите (Model) са интерфейси за достъп до базата данни/ записаните файлове. През тях се извършват действия като извличане на съдържание на поискана от директория/ създаване на потребител/ извличане на информация за потребителите/ извличане на информация за файловете.

Контролерите(Controller) управляват обработката на заявки към системата. Това са действия като: валидация на полета при попълване на форма, качване на файл, регистрация, логин, преглед на файл.

Изгледите (View) предоставят информация към потребителя в html формат. В тях се генерират html файлове за различни цели: форма за регистрация/ изглед към файлове в една директория/ меню за навигация.

Всичко това бива комбинирано от файла index.php, който в зависимост от параметъра на заявката ?route се избира, които изгледи да бъдат включени.

## 4. Използвани технологии

1. Използван е XAMPP за основна технология.
2. **СУБД:** MySQL
3. **PHP Server:** Apache
4. **Front end:** **html** страници, стилизирани със **css**. Без **javascript**.

## 5. Инсталация и настройки

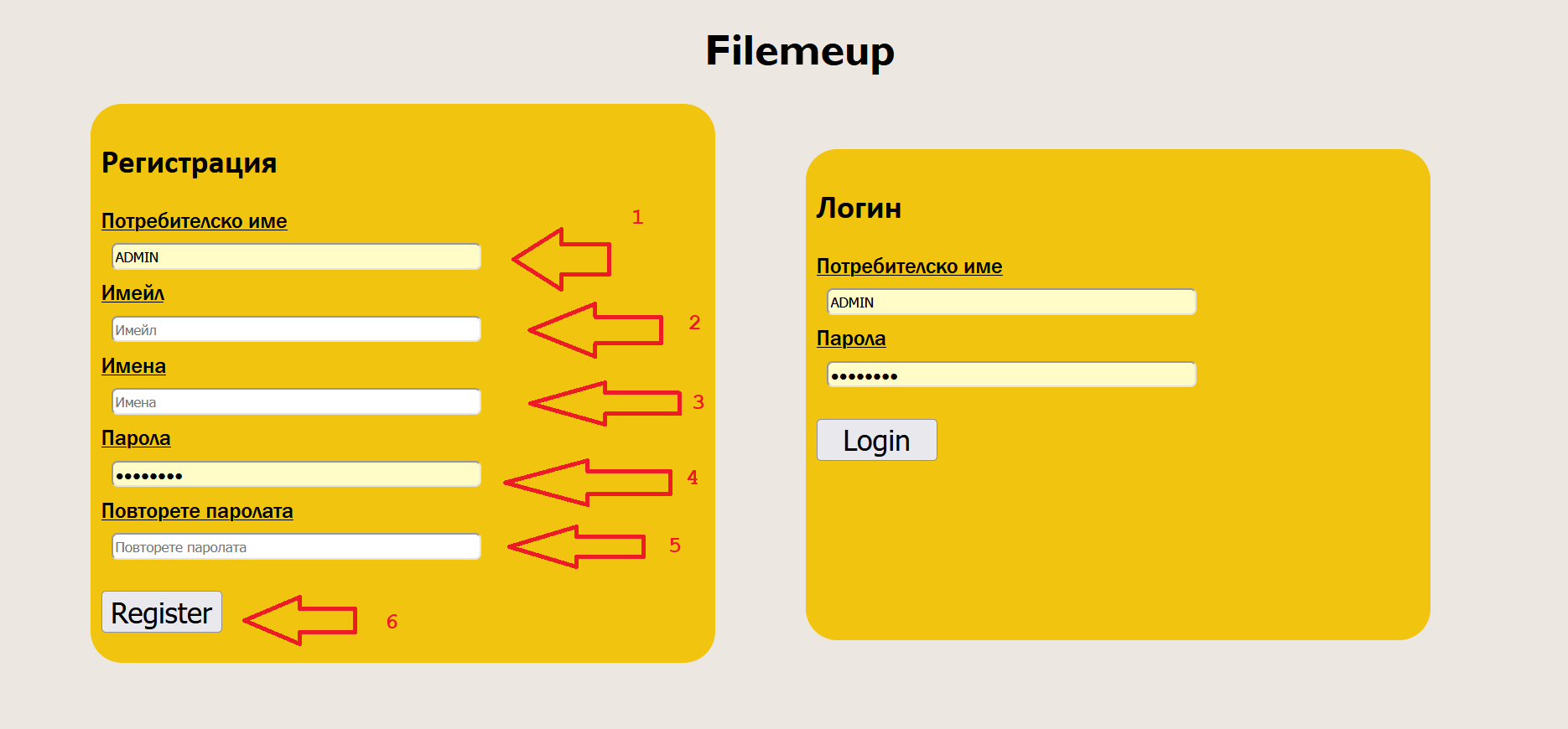
1. Архивът се разархивира и директорията filemeup се копира в директорията C:\\xampp\httdocs
2. Конфигурационните файлове за xampp се намират в filemeup\setup
3. SQL script за създаване и запълване с тестови данни на базата данни: filemeup\setup\db\_create.sql
4. Директорията filemeup\tmp\_init съдържа началното съдържание на директориите на двата тестови потребителя. Тя трябва да бъде копирана в директорията filemeup\tmp
5. Във файла filemeup\config.php се намират конфигурационни променливи за:

* Базов url на уебсайта
* Базова директория за уебсайта
* Информация за създателя, за уебсайта
* Потребителско име и парола за създаване връзка с базата данни

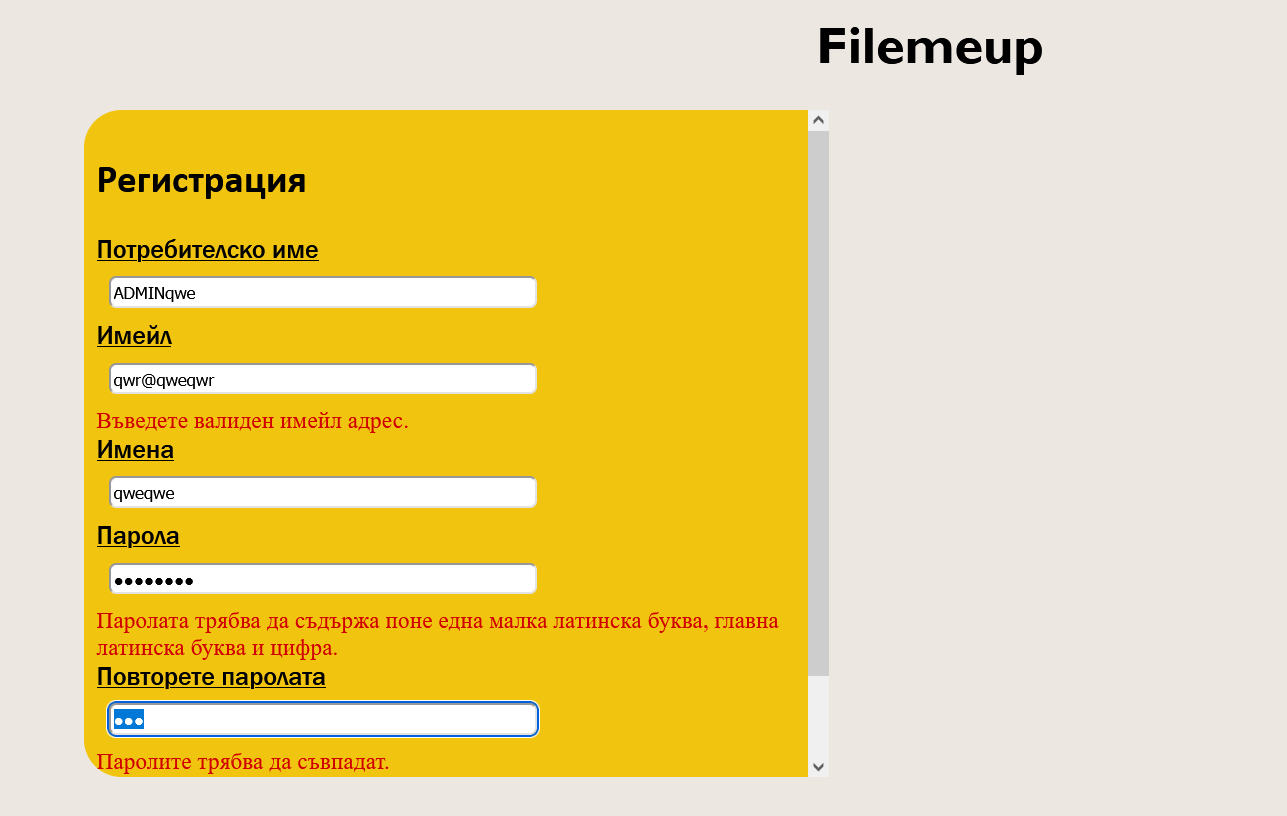
## 6. Кратко ръководство на потребителя

1. Регистрация

Попълнете формата за регистрация. При наличието на грешки – изпълнете инструкциите, които ще се покажат на екрана за да бъдат валидни попълнените данни.

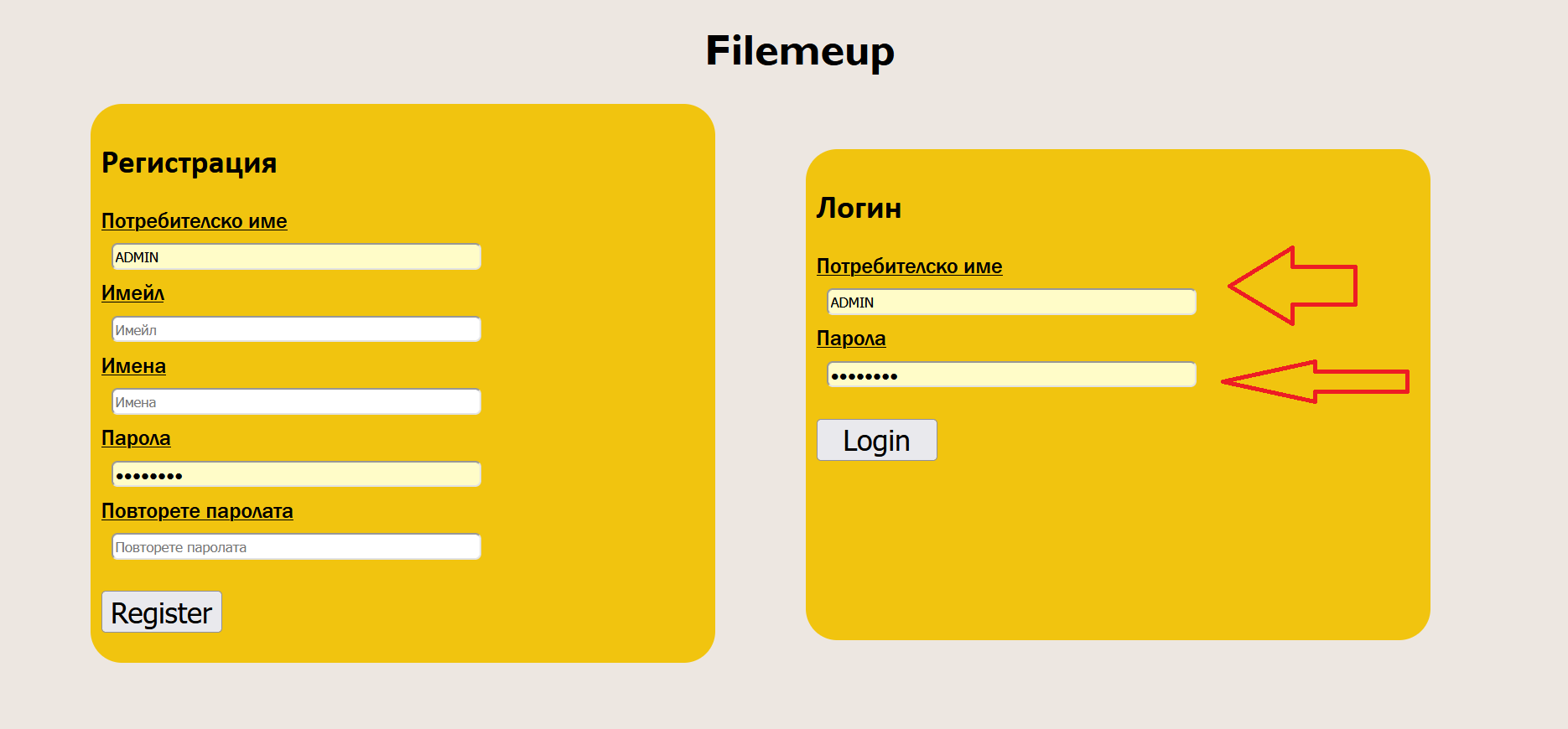


Пример за невалидно попълнена форма

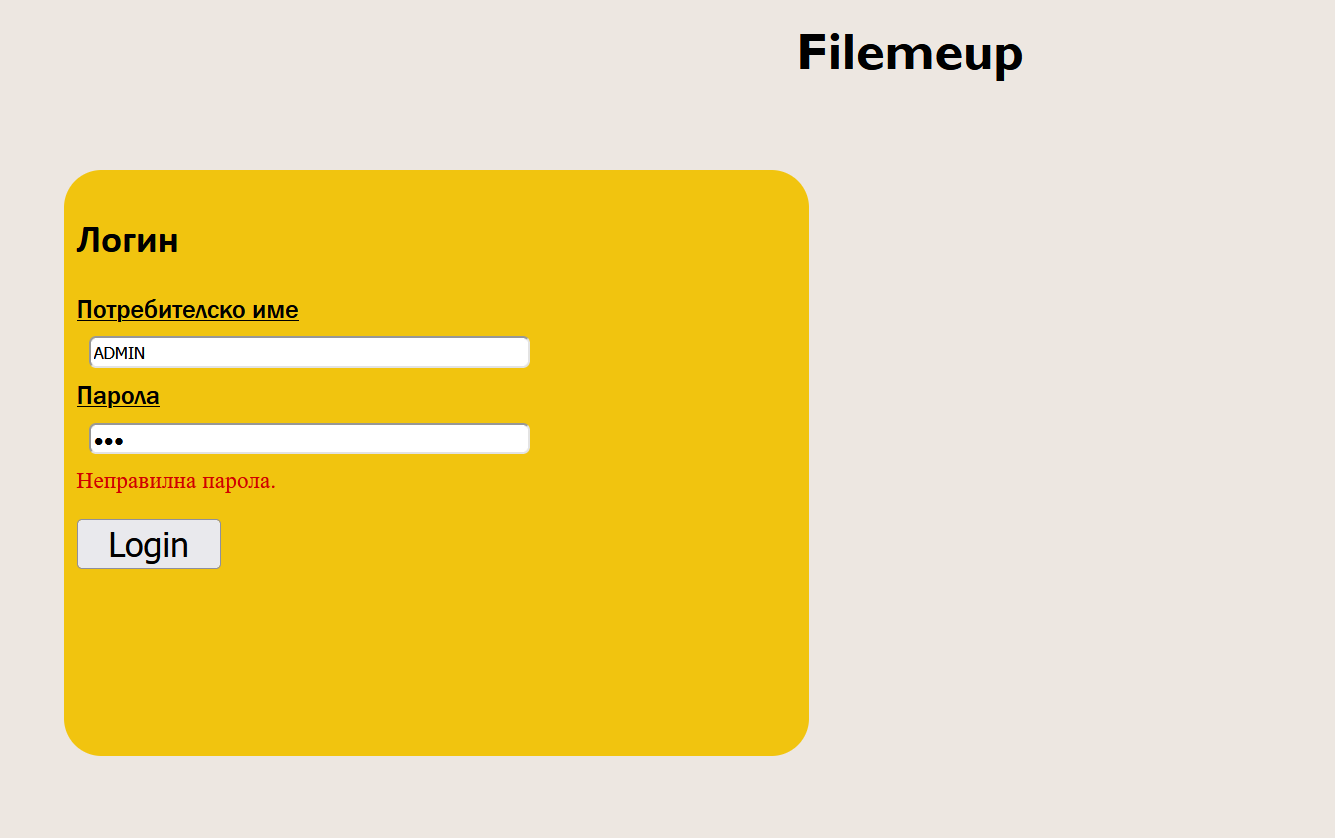


1. Логин

Попълнете формата за логин. При наличието на грешки – оправете входните данни.

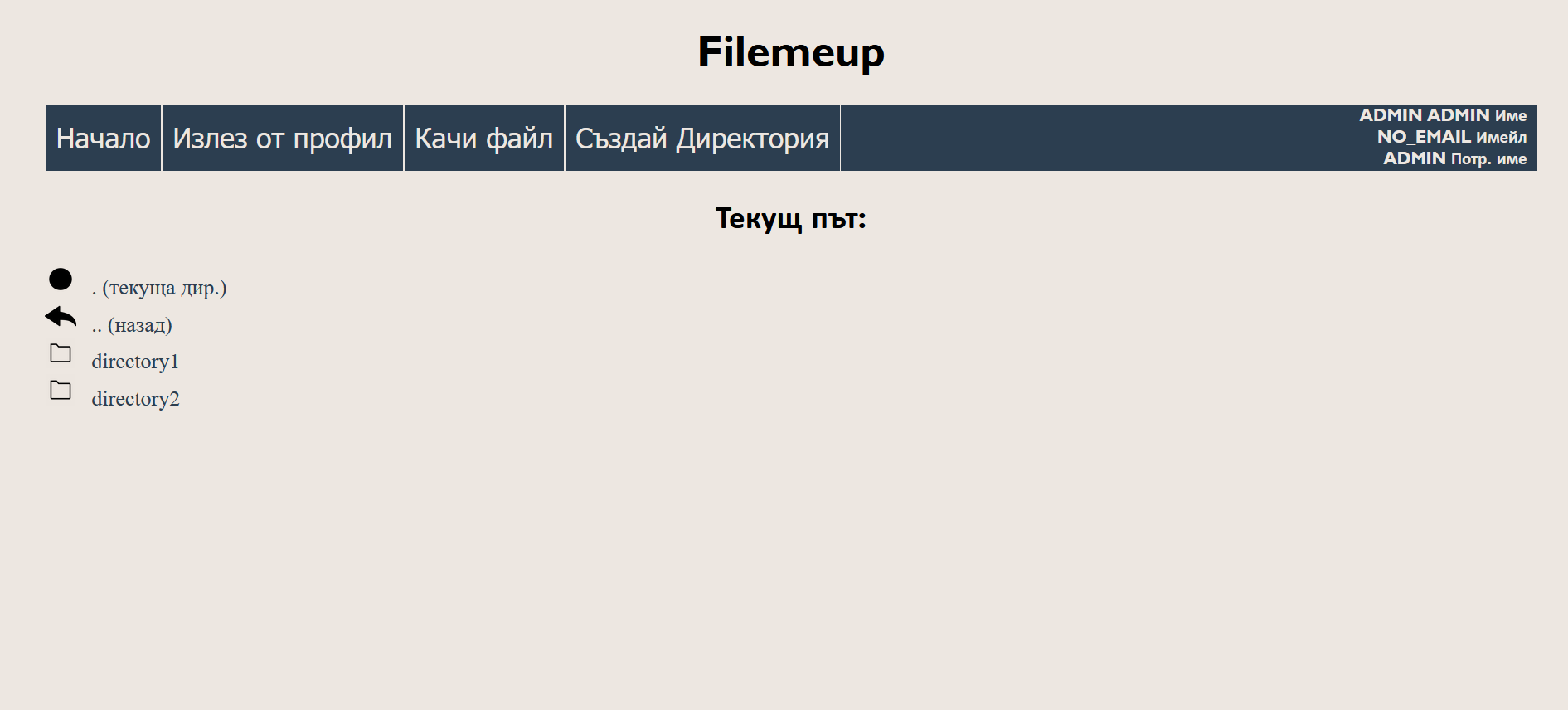


Грешка при логин:



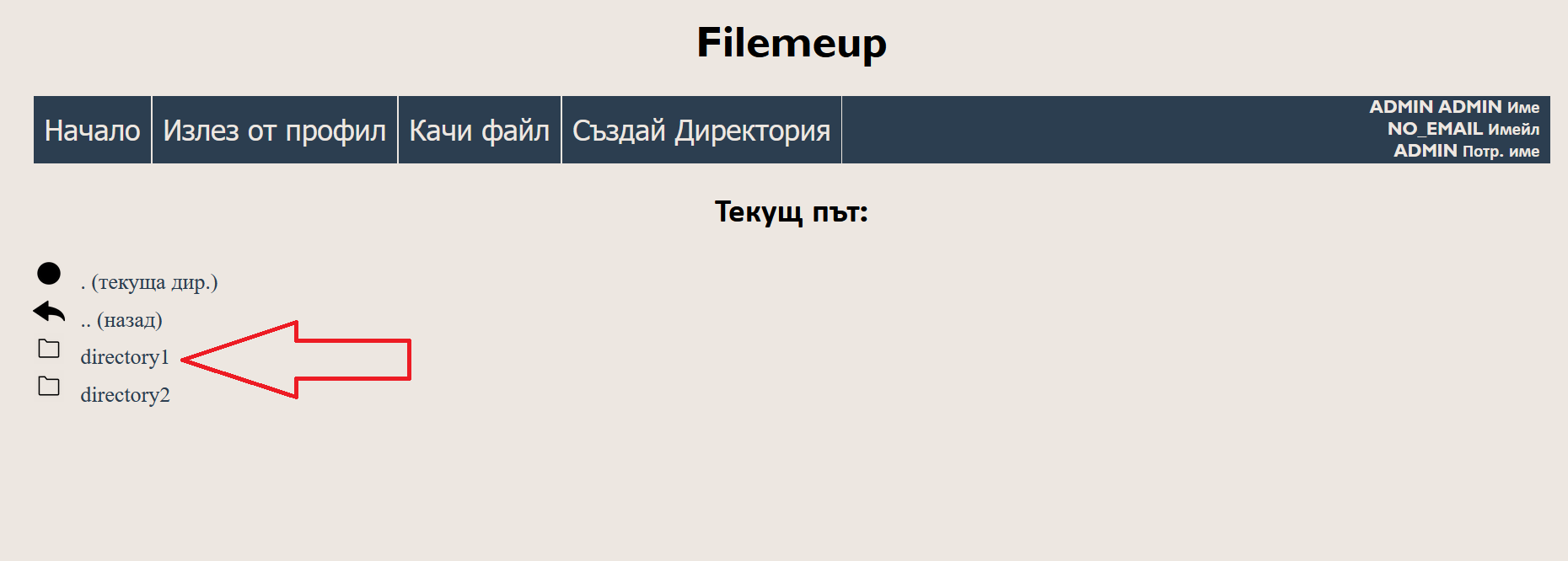
1. Изглед на директория

При успешно влизане в профила – показва се изгледа на директория. Там ще видите всички качени до момента файлове. След регистрация тази директория ще е празна.

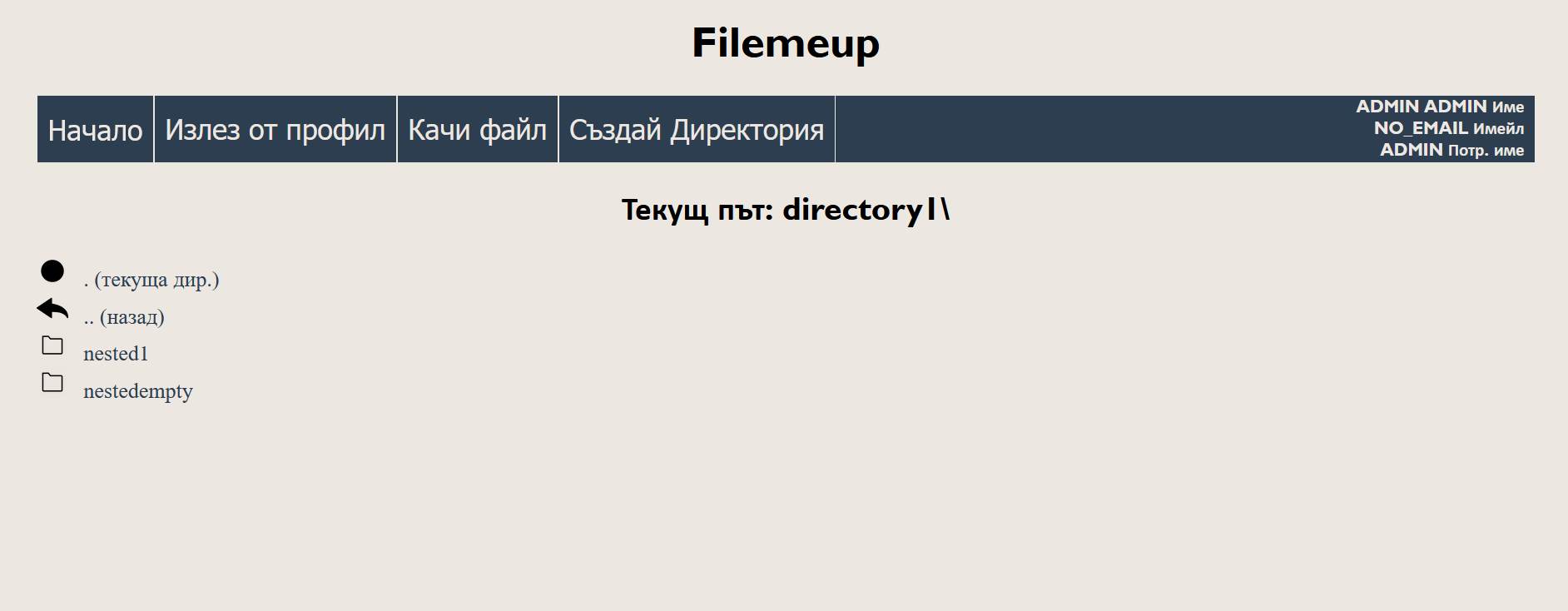


1. Навигация в директория

* Натиснете върху директория за да влезете в нея.



Резултат:

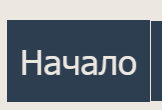


Различните видове файлове имат икони, представящи типа им.

* Избор на родителска директория

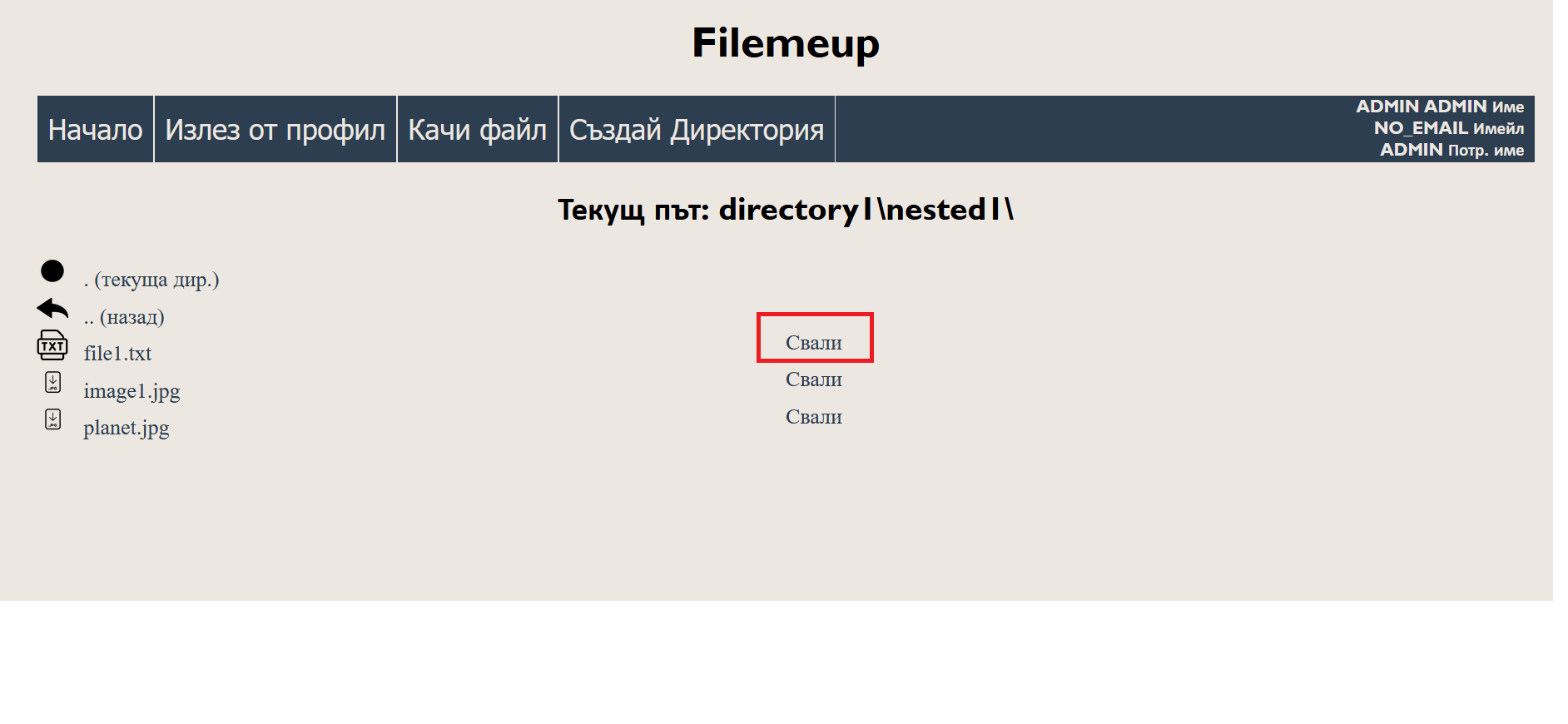


* Избор на начална директория



* Сваляне на файл

При наличието на файл, който не е директория – има възможност за неговото изтегляне.

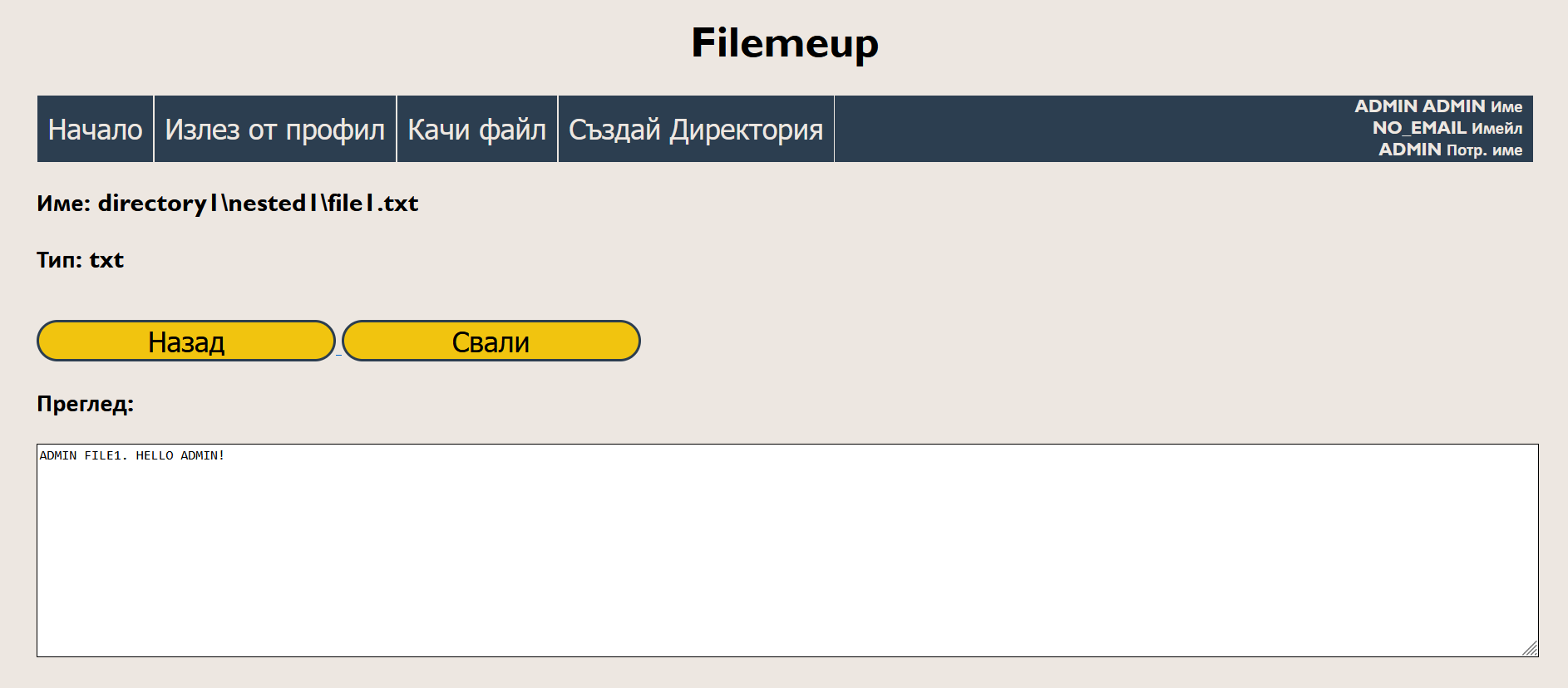


1. Изглед към файл

При избор на името на файл – показва се изглед към конкретния файл. Прави се опит за преглед на файла.



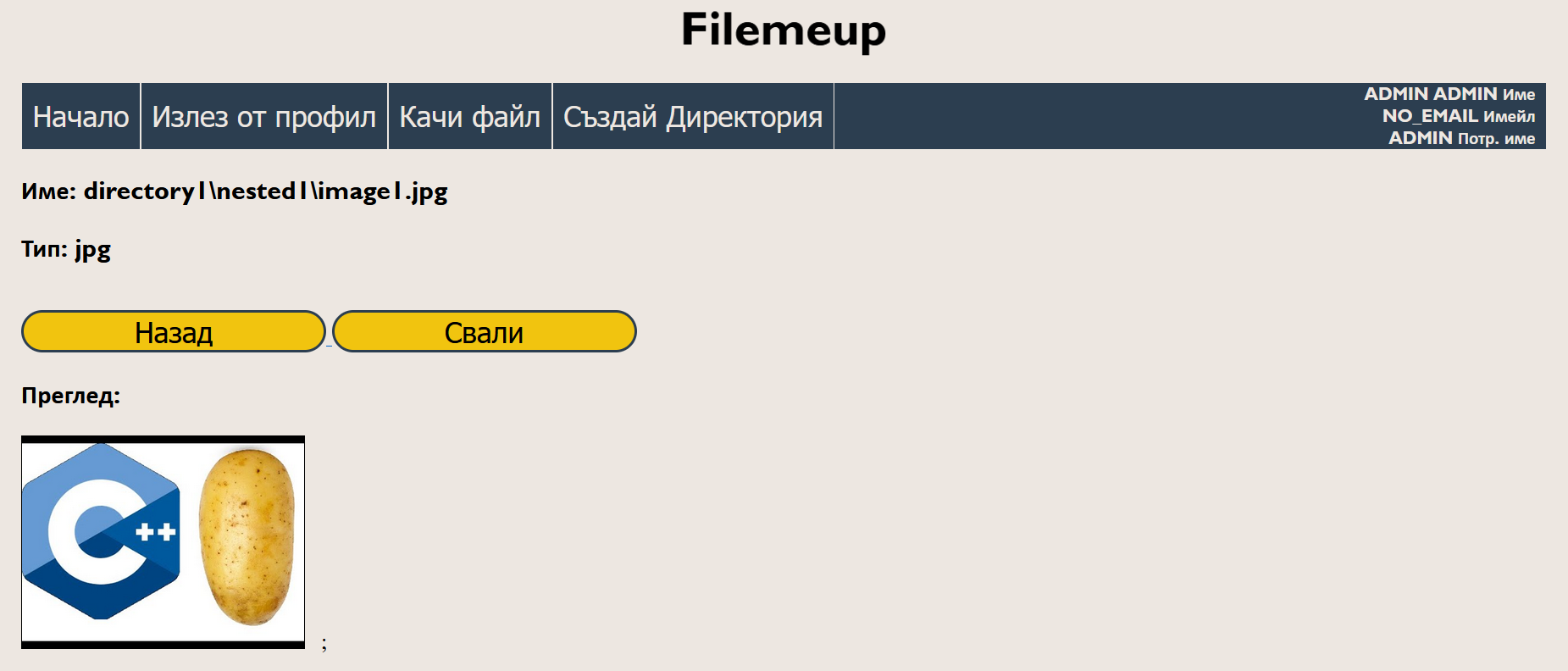
Резултат:



* Пример 2:

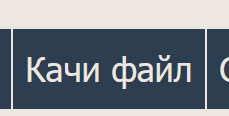


Резултат:



1. Качване на файл

* Изгледа за качване на файл се отваря със следния бутон от навигацията:



Резултат:

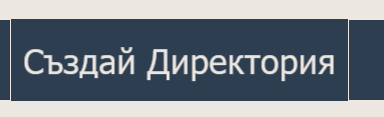


Тук потребителя може да избере файл, който да качи и неговото име. Ако полето за името остане празно, то файлът ще се запази с името, с което е качен.

След качване – потребителя бива пренасочен в директорията, в която се намира новия файл.

1. Създаване на директория

* Изгледа за създаване на директория се достъпва чрез следния бутон

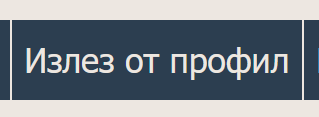


Резултат:



При попълване на валидно име – създава се директория в директорията, в която потребителя се е намирал, при натискане на бутона от менюто за навигация

1. Излизане от профила – потребител може да излезе от профила си със следния бутон:



## 7. Примерни данни

Създадени са 2 тестови потребителя:

* Username: ADMIN
* Password: ADMIN123

Има роля Admin и роля User.

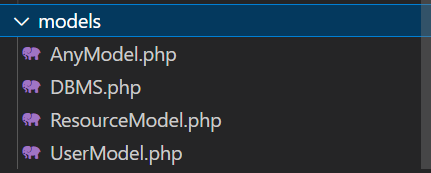
* Username: TEST
* Password: TEST123

Има роля User.

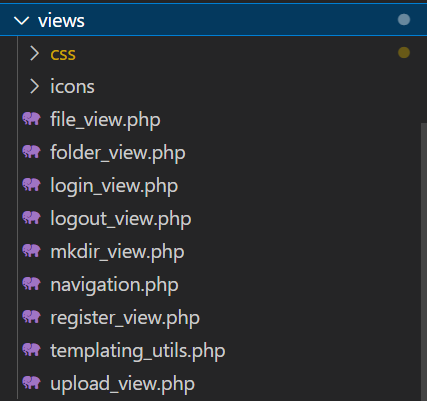
## 8. Описание на програмния код

3 основни директории:

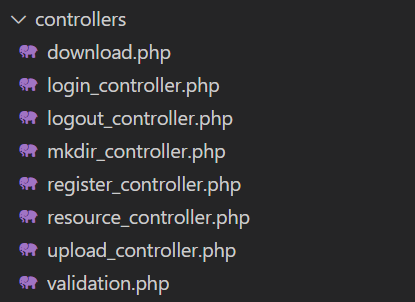
* Model
* View
* Controller



* **AnyModel.php –** абстрактен статичен клас, който бива наследяван от другите модели.Има базови функции за заявки към базата данни.
* **DBMS.php –** осъществява връзка с базата данни и предоставя достъп до нея
* **ResourceModel.php –** функции, работещи с файлове и директории (ресурси).
* **UserModel.php –** функции, свързани с извличане и запазване на информация, свързана с потребителите

****

* **register\_view/ login\_view – изгледи за регистрация и логин**
* **file\_view –** изглед към единичен файл
* **folder\_view** – изглед към директория
* **logout\_view –** бутон за излизане от профила
* **mkdir\_view** – изглед за създаване на директория
* **navigation –** навигационния панел, съдържащ главните бутони
* **templating\_utils –** помощни функции за създаване на форми
* **upload\_view –** изглед за качване на файл

****

* **download.php** – php script за изтегляне на файл от потребителя
* **login\_controller/ register\_controller –** логиката за регистрация и логин
* **logout\_controller –** логика за logout
* **mkdir\_controller** – контролер за създаване на директория
* **resource\_controller –** контролер за изгледите, които боравят с директории/ файлове
* **upload\_controller –** качване на файл от потребителя
* **validation.php** – помощни функции за валидация на данни от форми
* **config.php –** конфигурационни данни
* **index.php –** индексен файл

При зареждането на индексен файл, в зависимост от параметъра на заявката **?route** се включват съответните изгледи към основната html страница.

Запазването на път към файл, когато се превключва между страници, става чрез **?path** параметъра на заявката. В зависимост от това каква е стойността на **?path**, при качване на файл – този файл ще бъде качен различна директория.

При наличието на грешки при валидация на форма – съобщенията за грешки автоматично се събират и се изпращат в **?errors** параметъра на заявката.

Генерирането на форми става автоматично, чрез наличието на **form\_fields** конфигурационни променлива, която съдържа информация за полетата на формата и техните атрибути, описания, стойности. Базовата функционалност е имплементирана в **views/templating\_utils.php**

Валидацията на форми става автоматично, чрез **validation\_data**, променлива, която пази имената на полетата и списък от функции за валидация и съобщения за грешка. Основната функционалност е в **controllers/validation.php**

## 9. Приноси на студента, ограничения и възможности за бъдещо разширение

* Възможност за избиране на индексен файл (избира се html файл и се получава линк за неговото достъпване). При посещения на линка – автоматично се зареждат файловете, реферирани от **html** файла.
* Възможност за администрация на потребители
* Възможност за създаване на шаблони за проекти

## 10. Какво научих

Много в областта на разработката на уеб приложения с **xampp**. Запознах се с **php**.

## 11. Използвани източници

* <https://www.php.net/manual/en/index.php>
* <https://www.w3schools.com>

## 12. Добавена функционалност след първата версия

* **автоматично разархивиране на zip архиви**

Потребителите имат възможно да изберат качен архив да бъде разархивиран автоматично и да бъде включен в структурата на виртуалната му директория.



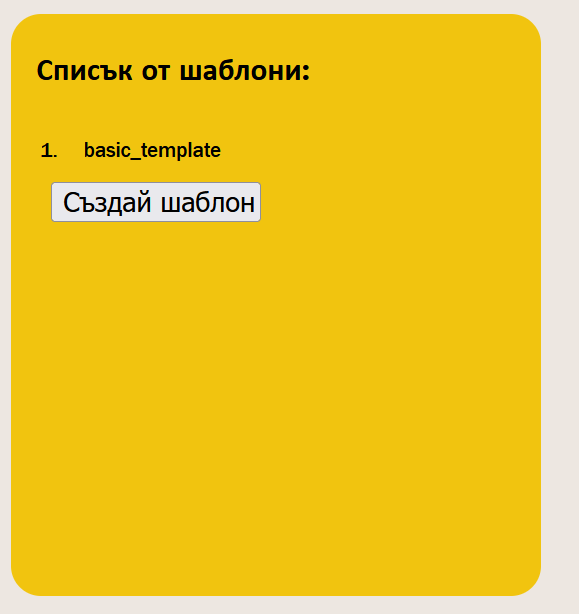
* **допълнително описание за всеки файл при изгледа на файловете**

Добавена е информация за размера и типа на файловете. Освен това е добавено меню за избор на шаблон, спрямо който да се тества някоя директория. Ако подструктурата на тази директория пасва на шаблона, то се изписва съответно съобщение на екрана.



* **възможност за създаване на шаблони**

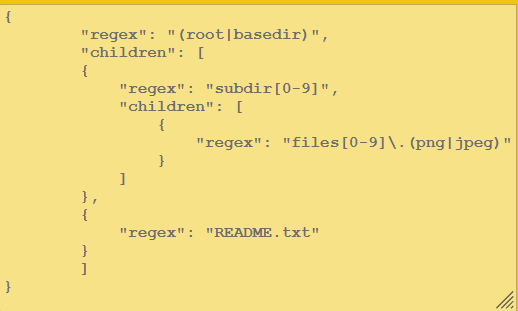
Създадена е функционалност за списък от всички създадени шаблони. Шаблон описва как да трябва да бъде структурирана една директория. Предоставена е функционалност за тестване дали една директория пасва на шаблон.



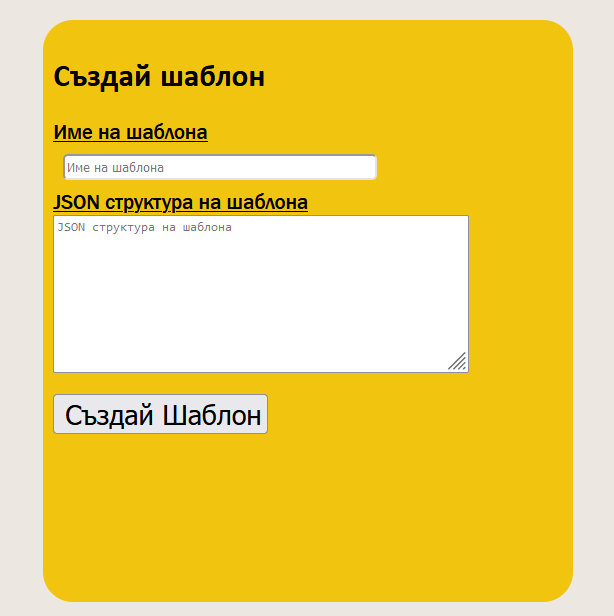
В менюто за създаване на шаблон се въвежда неговото име и освен това се въвежда и JSON текст, който описва шаблона в следния формат:

* + на ниво директория/ файл има regex, който описва какво може да бъде името на файла/директорията **(regex)**
  + всяката директория може да съдържа много обекти за директории, които съответно си имат в масива **(children).** Всяко дете се получава по описаната схема
  + Ако един обект има **regex**, но няма **children**, то той описва файл, а не директория

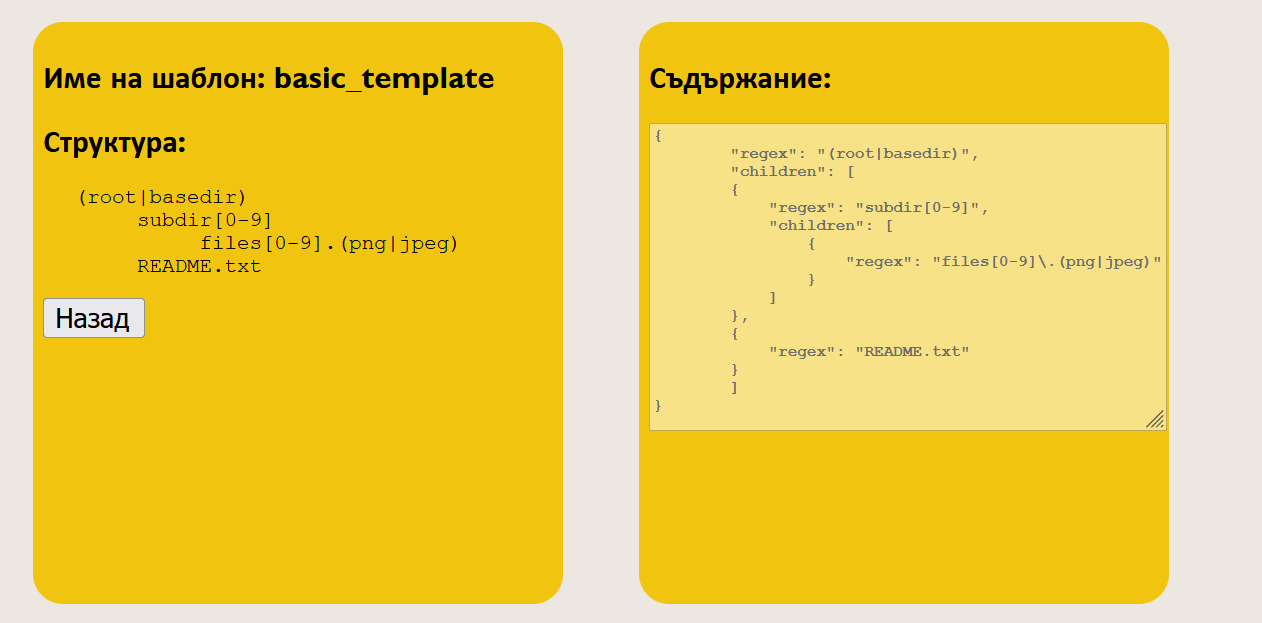
Примерна структура:



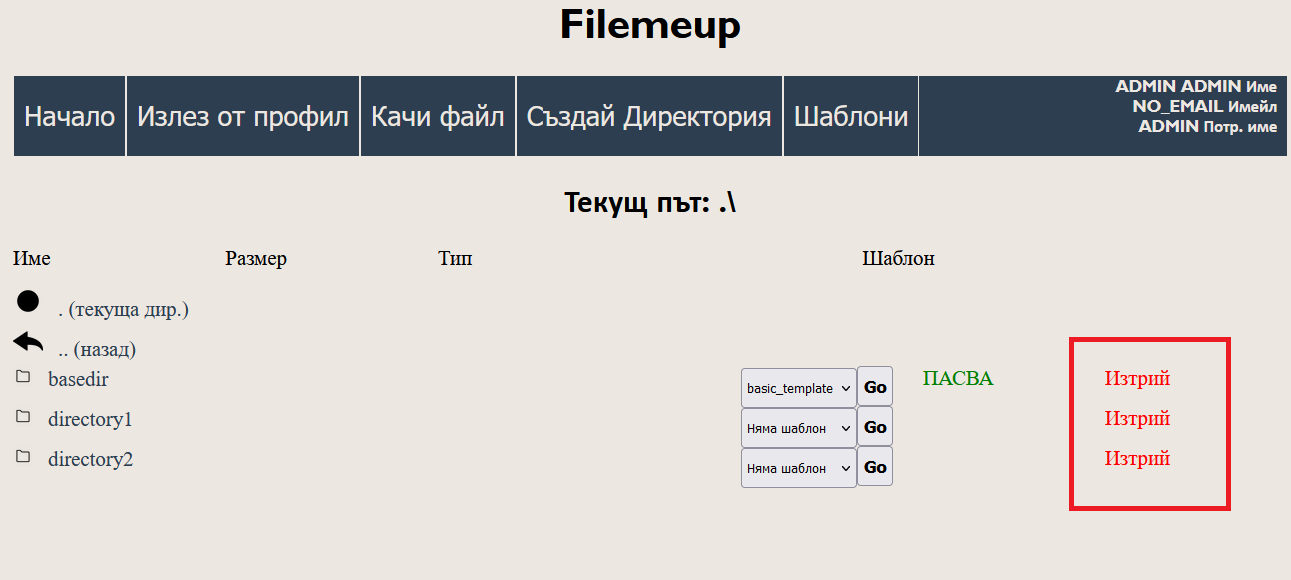
Меню за създаване на шаблон:

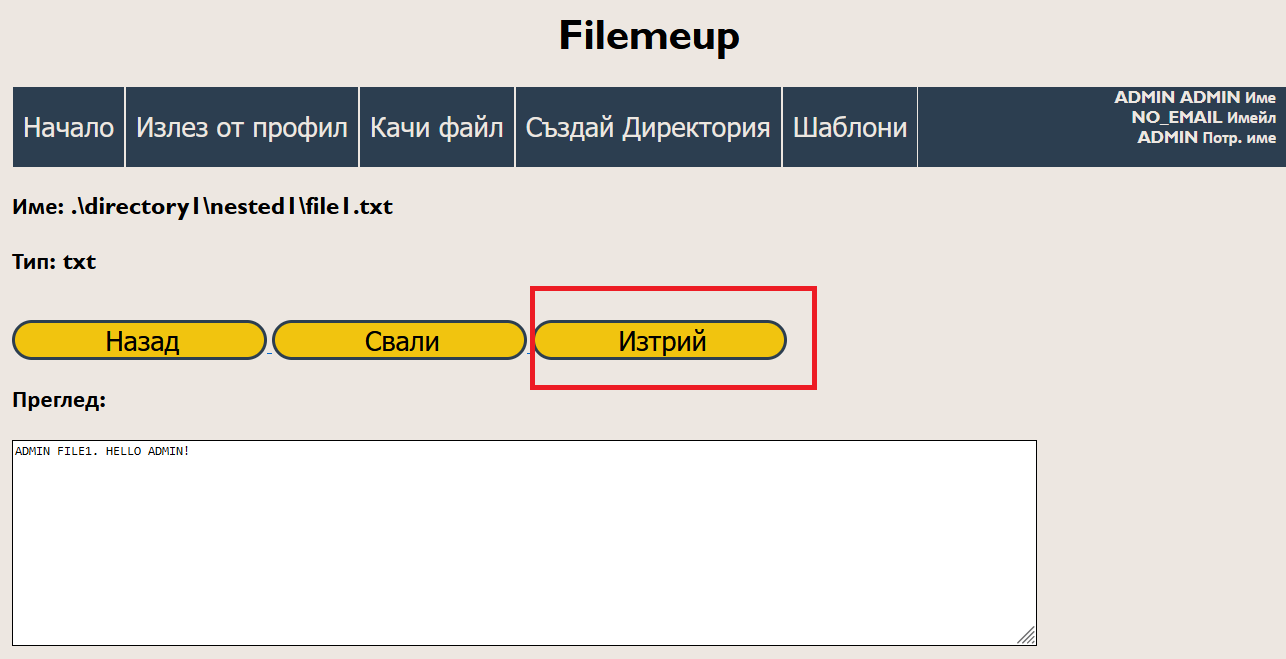


Меню за разглеждане на шаблон:



* функционалност за изтриване на файлове – възможно е файл или директория да бъде изтрит при натискане на бутона **Изтрий** през изгледа на директории или изглед на конкретен файл





* config\_validation.php с конфигурация на валидациите на форми
* функционалност за валидация на подструктура на директория според някакъв шаблон. След избиране на съответен шаблон и натискане на Go се извършва проверка върху избраната директория. Проверява се дали подструктурата на директорията пасва на шаблона ( дали всички файлове, които са в директорията имат описание в шаблона).





Предал (подпис): ………………………….

/*Димитър Татарски, КН, 4курс, 81818, 7 група*/

Приел (подпис): ………………………….

/доц. *Милен Петров*/