Лабораториска вежба 1

Автентикација

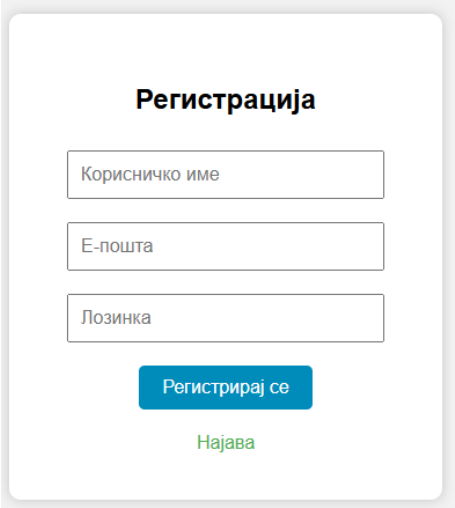
За изработка на оваа лабораториска вежба, користев чист Java код, без употреба на готови frameworks кои помагаат за автентикација, хеширање и слично, туку направив свои класи кои мануелно го прават тоа.

Апликацијата која ја изработив е едноставен веб сервер кој користи едноставен HTTP сервер, библиотека без употреба на Spring Boot и слични библиотеки како тоа. Оваа библиотека обработува HTTP барања како што се GET и POST и креира контексти (URL патеки).

Основни функционалности на апликацијата се :

1. Регистрација на корисници (/site/signup)
2. Најава на корисници (/site/login)
3. Dashboard за најавени корисници (/site/dashboard)
4. Одјава на корисници (site/logout)
5. File users.txt

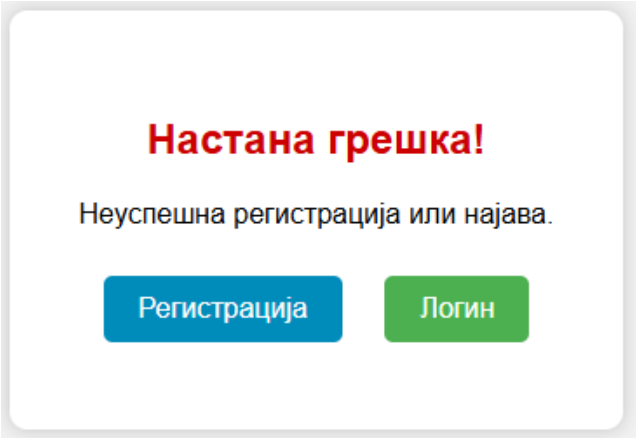
1. Регистрација на корисници



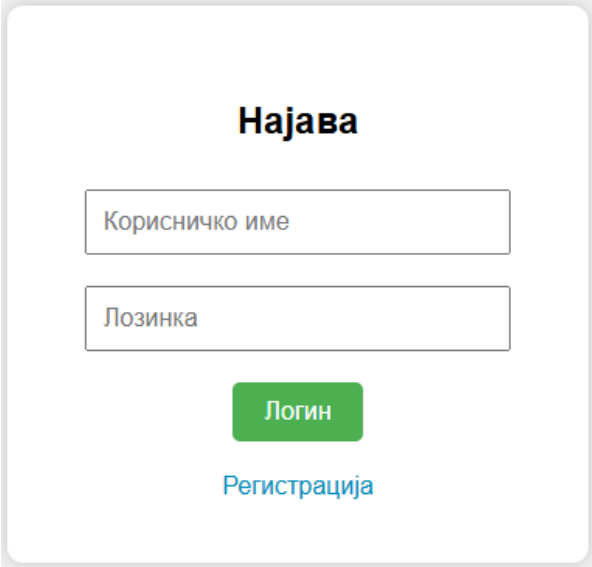
На формата за регистрација, корисникот внесува корисничко име, e-mail и лозинка.
Апликацијата проверува дали корисничкото име или e-mailот веќе постојат, дали некој ги искористил, тоа се прави со методот userExists.

Лозинката се хешира со SHA-256 и случаен salt, а податоците се запишуваат во .txt документ, кој во случајот служи како мала база на податоци кој ги чува корисничкото име, e-mail, хеширана лозинка и salt.
За појаснување, на лозинката, пред таа да се хешира, и се задава рандом salt вредност, па потоа следи хеширањето со SHA-256. Тоа го направив со цел да нема двајца корисници со иста лозинка, а во база да им се чуваат лозинките со иста вредност по хеширањето, како што е случајот со чист SHA-256.
Но, важно е дека SHA-256 ја хешира лозинката 50 000 пати.
Доколку регистрацијата е успешна, корисникот се пренасочува кон страницата за најава (/site/login).

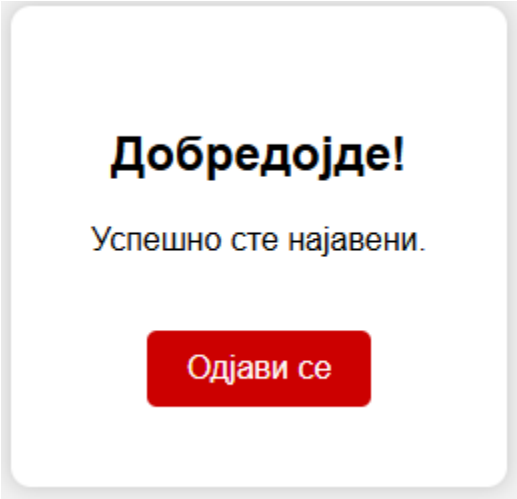
Но, доколку има грешка, корисникот е пренасочен на fail.html



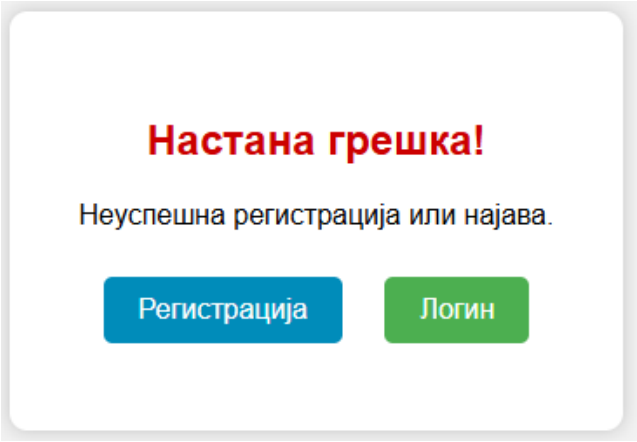
2. Најава на корисници



При најава, корисникот внесува корисничко име и лозинка. Потоа апликацијата проверува во users.txt каде се чуваат корисничкото име, e-mail, хеширана лозинка и salt, за тоа дали постои корисник со тоа корисничко име и лозинка, а лозинката се проверува: ако хешот на внесената лозинка се совпаѓа со зачуваниот хеш, најавата е успешна.
При успешна најава, корисникот е пренасочен на /site/dashboard. Покрај тоа, се генерира токен кој се испраќа како cookie до корисникот и истиот работи со HttpOnly.
Ова спречува било каков client-side JavaScript код да пристапи до cookie-то и да се украдат информациите за сесијата. Со тоа, се постигнува критична заштита од Cross-Site Scripting напади, затоа што напаѓачот нема да може да украде сесиски токен преку инјектирање на скрипта.



Ако најавата не е успешна, се прикажува fail.html со порака за грешка.



3. Dashboard за најавени корисници (/site/dashboard)

Овде корисникот може да пристапи само при успешна најава. При секое барање се проверува дали cookie со токен постои и дали токенот е валиден во Session Manager. Доколку сесијата е валидна, се прикажува dashboard.html.
Но, ако сесијата не е валидна или токенот не постои, корисникот се пренасочува кон страницата за најава.

4. Одјава на корисници (site/logout)
   При клик на “Одјави се”, токенот од cookie се брише од SessionManager, а корисникот е пренасочен кон /site/login. Овој процес го прекинува пристапот до dashboard и ја завршува сесијата.
5. File users.txt
   Фајлот users.txt служи како едноставна база на податоци за зачувување на кориснички информации. Секој корисник е запишан во еден ред во формат:
   - **username** – корисничкото име на корисникот
   - **email** – е-пошта на корисникот
   - **passwordHash** – хеширана лозинка, резултат од SHA-256 применет на лозинката заедно со salt
   - **salt** – случајна низа која се додава на лозинката пред хеширање за зголемување на безбедноста

   На овој начин, дури и ако двајца корисници имаат иста лозинка, поради различниот salt нивните хеширани лозинки ќе бидат различни.