

Веб апликација за шах која овозможува трансформација на вистинска слика од шаховска табла во дигитален формат и евалуација на позицијата со помош на шаховски алгоритам

Изработил

Бојан Робев

Содржина

[1 Вовед 3](#_Toc157470192)

[2 Искористени технологии 5](#_Toc157470193)

[2.1 Python (Django Framework) 5](#_Toc157470194)

[2.2 Node.js 6](#_Toc157470195)

[2.3 React 7](#_Toc157470196)

[2.3.1 Babel 7](#_Toc157470197)

[2.3.2 Webpack 7](#_Toc157470198)

[2.3.3 Axios 7](#_Toc157470199)

[2.3.4 React Router Dom 8](#_Toc157470200)

[2.3.5 Chess.js 8](#_Toc157470201)

[2.3.6 React Chessboard 8](#_Toc157470202)

[2.3.7 Material UI 8](#_Toc157470203)

[2.4 Stockfish 8](#_Toc157470204)

[2.5 Keras 8](#_Toc157470205)

[2.6 Pipenv 8](#_Toc157470206)

[3 Детален опис на веб апликацијата 9](#_Toc157470207)

[3.1 Почетна страна 9](#_Toc157470208)

[3.2 Страна за внесување на слика 9](#_Toc157470209)

[3.3 Страна за промена на шаховската позиција 11](#_Toc157470210)

[3.4 Главна страна 11](#_Toc157470211)

[3.4.1 Функција за евалуација на шаховска позиција 13](#_Toc157470212)

[3.5 Можност за навраќање 15](#_Toc157470213)

[4 Структура на проектот 16](#_Toc157470214)

[4.1 Главна структура 16](#_Toc157470215)

[4.1.1 Бекенд 17](#_Toc157470216)

[4.1.2 Иницијализирање на Django 18](#_Toc157470217)

[4.1.3 Фронтенд 18](#_Toc157470218)

[5 Модели за препознавање 19](#_Toc157470219)

[5.1 Модел за препознавање шаховска табла 19](#_Toc157470220)

[5.2 Модел за препознавање шаховски фигури 21](#_Toc157470221)

[5.2.1 Готов модел (VGG16) 21](#_Toc157470222)

[5.2.2 Податочно множество 22](#_Toc157470223)

[5.2.3 Структура на мрежата 22](#_Toc157470224)

[5.2.4 Тренирање на моделот 22](#_Toc157470225)

[5.2.5 Резултати од тренирање на моделот 23](#_Toc157470226)

[6 Можности за проширување на веб апликацијата 24](#_Toc157470227)

[7 Заклучок 24](#_Toc157470228)

Абстракт

Во рамките на овој дипломски труд ќе биде опфатен целосниот дизајн на веб страна со интегрирани модели за машинска визија со цел да се напарави интерактивна, брза и модерна веб страна која инкорпорира техники од веб програмирање и машинско учење. Целта на проектов е да се даде начин новите шахисти да имаат брз и едноставен начин на анализа на шаховска позиција со помош на моќниот Stockfish шаховскиот алгоритам, така што со интуитивен и минималистички кориснички интерфејс ќе им помогне за ефикасно надоградување на шаховските вештини на корисникот.

# **1 Вовед**

Во рамките на овој дипломски труд, се изработува целосна веб апликација со цел да се обезбеди на шахистите почетници вреден софтвер наменет за почетници и професионалци. Веб апликацијата овозможува анализа на шаховски позиции во реално време и дава совети за подобрување на потега користејќи го еден од помоќните шаховски алгоритми нарачен Stockfish.

Целта на овој проект е да ги научи шахистите почетници да ги разбираат и анализираат шаховските позиции, како и да ги води кон продлабочено мислење на . Со употреба на алгоритми за машинско учење и машинска визија, нашата апликација овозможува трансформација на слика од физичкиот шах во дигитален формат и евалуација на шаховската позиција.

Преку три различни начини за внес на шаховската позиција, корисниците можат да ја стартуваат анализата. По праќањето на сликата до бекендот, апликацијата резултира со FEN (Forsyth-Edwards Notation) формат од шаховската позиција. Сепак, бидејќи не може да се постигне 100% прецизност на не тривијални слики, направена е функционалност за рачна корекција на детектираните грешки. Потоа, корисниците имаат можност да внесат дополнителни информации за состојбата на играта, како потенцијални рокади и кој играч е на ред, и да го изградат целосниот FEN се’ со цел успешно да се валидира од страна на Stockfish. Овие дополнителни информации се критични за градење на FEN бидејќи Stockfish шаховскиот алгоритам не работи во случај да се внесе непостоечка шаховска конфигурација или комбинација.

По успешното внесување и корекција на шаховската конфигурација, корисниците се пренасочуваат кон страницата за евалуација на шаховската позиција. Тука, апликацијата нуди графички приказ на најдобрите 3 потега на играчот кој е на потег, како и информации за моменталното водство помеѓу играчите. Дополнително, после секој потег, се ажурираат стрелките за 3те најдобри потега и се прави повторна евалуација на шаховската позиција.

Со овој проект, го обезбедиме шахот како забавна и достапна игра за почетници, истовремено обезбедувајќи инструмент за подобрување на нивните шаховски вештини. Преку комбинирање на технологии од областите на веб дизајн, машинско учење и машинска визија, создадена е платформа која овозможува раст на шаховските способности кај корисниците.

Во продолжение на овој дипломски труд, ќе ги истражиме деталите на нашиот проект, вклучувајќи го описот на проблемот, технологиите кои ги користиме, архитектурата на бекендот и фронтендот, моделот за препознавање на краевите на шаховската табла, моделот за препознавање на фигурите, сликовит описот на целата веб апликација и можности за проширување на истата.

# **2 Искористени технологии**

## 2.1 Python (Django Framework)

Django е високо-ниво Python веб-рамка за брзо развивање на сигурни и одржливи веб-страници. Django следи филозофијата "Batteries included" и обезбедува готово сè што развивачите можат да сакаат. Django ја следи архитектурата Model-View-Controller (MVC) за структурирање на веб-апликации. Во оваа шема, 'Model' се однесува на делот кој управува со базата на податоци, 'View' обработува барања и враќа одговори, додека 'Controller' управува со интеракцијата меѓу моделот и приказот. Django исто така интегрира ORM (Object-Relational Mapper), кој овозможува ефикасна интеракција со различни бази на податоци преку објектно-ориентиран пристап. Шаблонскиот јазик на Django овозможува флексибилно и динамичко создавање на HTML документи. Покрај тоа, Django нуди вградена поддршка за управување со сесии и автентификација на корисници.

Middleware во Django е рамка за поврзување со процесот на обработка на барања/одговори. Django Middleware се состои од слоеви на софтвер кои се извршуваат за време на барањето/одговорот на веб-пребарувачот во Django апликацијата. Тие функционираат како хуки за глобална обработка на барањата, овозможувајќи предобработка и послеобработка на HTTP барањата и одговорите.

Секој компонент на Middleware е одговорен за специфична функција и може да биде прилагоден или проширен според специфичните потреби на апликацијата. Некои од Middleware компонентите во Django вклучуваат:

* AuthenticationMiddleware: Управува со процесот на автентификација.
* SessionMiddleware: Управува со сесиите на корисниците.
* CsrfViewMiddleware: Обезбедува заштита од CSRF напади.
* SecurityMiddleware: Управува со безбедносните поставки.

## 2.2 Node.js

Node.js е популарен open-source, cross-platform JavaScript runtime околина која овозможува извршување на JavaScript код надвор од веб прелистувач. Node.js користи V8 JavaScript engine, истиот што го користи и Google Chrome, за да овозможи брзо извршување на JavaScript код.

Преглед на Клучни Карактеристики

* **Асинхроност и Non-Blocking I/O**

Node.js користи асинхронен, non-blocking влезно-излезен модел, кој овозможува висока пропусна моќ и ефикасна употреба на ресурсите. Ова е особено важно за апликации кои се соочуваат со голем број на истовремени барања, бидејќи овозможува обработка на повеќе операции во исто време без блокирање на главната нишка.

* **Event-Driven Architecture**

Системот на Node.js е целосно ориентиран кон настани (events), користејќи го EventEmitter класата за да овозможи објекти да емитуваат настани кои потоа можат да бидат слушнати и обработени. Оваа архитектура е особено корисна за развој на комплексни апликации каде што одговорот на различни системски настани е критичен.

* **npm - Node Package Manager**

Node.js е поддржан од npm, кој е глобален репозиториум за споделување и управување со JavaScript библиотеки. Npm овозможува развивачите лесно да ги инсталираат, ажурираат и управуваат со зависностите на нивните проекти, што значително ја зголемува ефикасноста на развојот.

* **Модуларност**

Node.js промовира модуларен пристап кон развојот на апликации, овозможувајќи организација на кодот во мали, управливи модули. Ова ја зголемува читливоста, одржувањето и тестирањето на кодот.

* **Cross-Platform**

Node.js може да се извршува на различни платформи, вклучувајќи Windows, macOS и Linux. Оваа крос-платформска компатибилност го прави идеален за развој на универзални апликации кои можат лесно да се поставуваат на различни околини.

* **Скалабилност**

Node.js е одличен избор за апликации кои очекуваат висока посетеност или интензивно користење на ресурсите.

* **Frameworks и Libraries**

Богатиот екосистем на рамки и библиотеки, како Express.js, KoaJS, NestJS, и други, овозможува развивачите да брзо и ефикасно изградуваат апликации, со значително намалување на потребата за пишување на код од нула.

## 2.3 React

React е модерна JavaScript библиотека која се користи за изградба на кориснички интерфејси, првенствено за веб апликации. Централниот елемент на React се компонентите, кои се независни, реупотребливи делови код кои дефинираат изгледот и функционалноста на деловите од корисничкиот интерфејс. Компонентите се клучни во создавањето на структуриран и модуларен код, овозможувајќи развивачите да изградат сложени интерфејси од помали, управливи делови.

Една од најзначајните карактеристики на React е користењето на JSX, синтакса која комбинира JavaScript со HTML елементи. Оваа синтакса олеснува создавањето на интерфејсите, овозможувајќи директно вметнување на HTML во JavaScript кодот. JSX ја прави кодната структура почитлива и поедноставна за разбирање, олеснувајќи ја реализацијата на динамички UI елементи.

Во срцето на React лежи управувањето со 'state' и 'props'. 'State' се однесува на сетот на податоци кои контролираат однесувањето на компонентата, додека 'props' се користат за пренесување на податоци и конфигурации на различни компоненти. Оваа дистинкција овозможува ефикасно управување со податоците и интеракцијата меѓу компонентите.

Животниот циклус на компонентите е уште еден критичен аспект на React. Секоја компонента во React поминува низ различни фази како што се монтирање, ажурирање и демонтирање.

### 2.3.1 Babel

Babel е JavaScript компајлер кој овозможува претворање на современ JavaScript код во формат кој е компатибилен со постарите веб прелистувачи. Ова е особено важно за поддршка на нови JavaScript функции кои можеби сè уште не се поддржани во сите прелистувачи. Babel исто така поддржува JSX за React апликации.

### 2.3.2 Webpack

Webpack е модуларен пакувач (bundler) за JavaScript апликации. Тој овозможува организирање и пакување на различни ресурси и модули (како JavaScript, CSS, и слики) во оптимизирани пакети за продукција. Webpack исто така поддржува трансформации и оптимизации на кодот преку различни додатоци (plugins).

### 2.3.3 Axios

Axios е популарна JavaScript библиотека за извршување HTTP барања. Таа се користи за воспоставување комуникација со сервери и API-ња.

### 2.3.4 React Router Dom

Оваа библиотека е стандард за декларативно рутирање во React апликации. React Router Dom овозможува дефинирање на рути (routes) во React апликациите, управување со навигацијата, и динамичко вчитување на компонентите во зависност од URL-то на прелистувачот.

### 2.3.5 Chess.js

Chess.js е компонента која ја обезбедува целосната шаховска логика и правила на играње. Таа нуди функции за валидација на потегата и следење на состојбата на играта.

### 2.3.6 React Chessboard

React Chessboard е React компонента која нуди визуелен интерфејс за шаховски табли. Оваа компонента може лесно да се интегрира со Chess.js за создавање на комплетна шаховска апликација, овозможувајќи приказ и интеракција со шаховската табла.

### 2.3.7 Material UI

Material UI е популарен сет на React компоненти кои следат принципите на Google-овиот Material Design. Тој нуди богат избор на предефинирани компоненти како копчиња, картички, дијалози и други корисни компоненти, кои овозможуваат брзо и ефикасно изградување на атрактивни и функционални кориснички интерфејси.

## 2.4 Stockfish

Stockfish е бесплатен и моќен шаховски алгоритам, кој е изведен од Glaurung 2.1. Тој е дизајниран за анализирање на шаховски позиции и пресметување на оптимални потези. Stockfish не вклучува графички кориснички интерфејс (GUI), но може да се користи со различни достапни GUI-а.

## 2.5 Keras

Keras е Python библиотека со отворен код, специјализирана за изградба на невронски мрежи. Таа нуди едноставен и пристапен начин за креирање на сложени модели за машинско учење, истовремено поддржувајќи различни постоечки бекенди како TensorFlow. Со фокус на лесноста на употреба и градење на модули, Keras овозможува брзо градење на прототип модели и експериментирање во областа на длабокото учење.

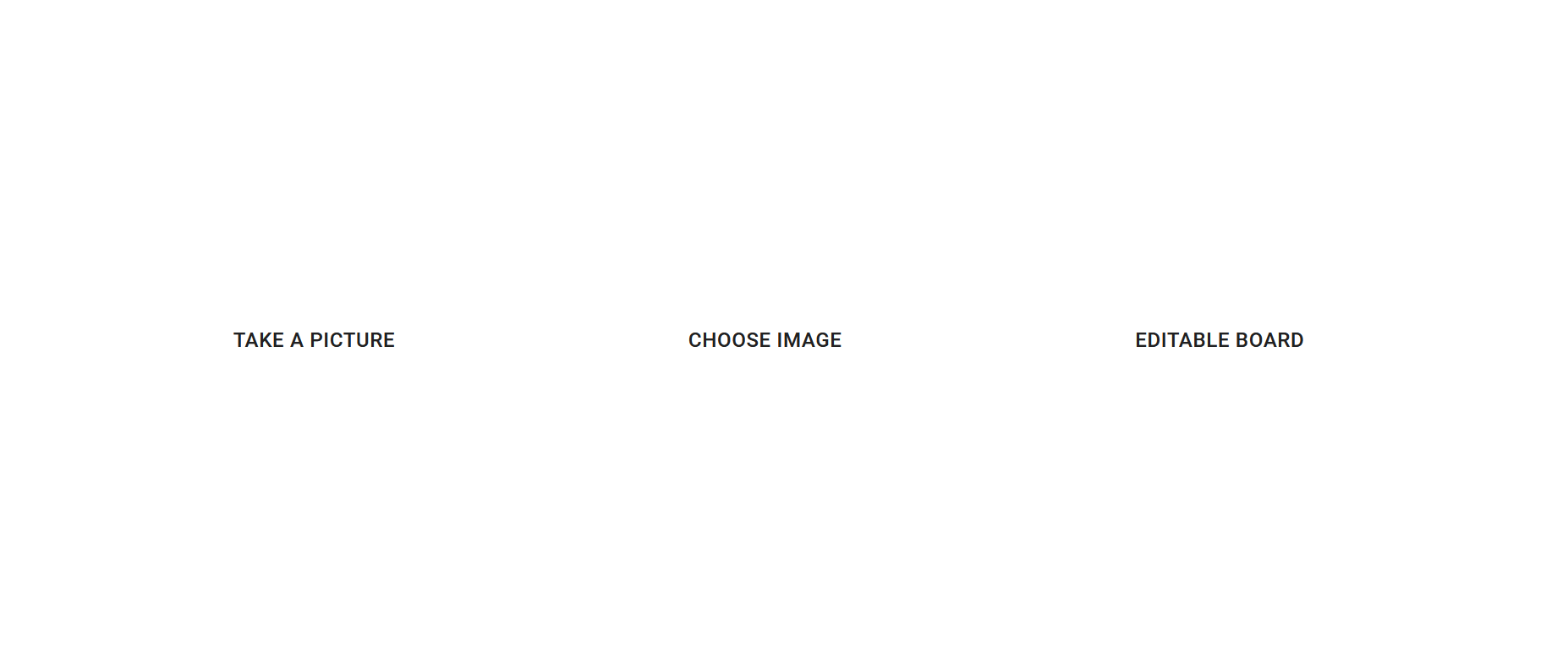
## 2.6 Pipenv

Pipenv е алатка која се користи во Python за управување со зависности и виртуелни околини. Таа комбинира Pip и virtualenv во едно, овозможувајќи лесно создавање на виртуелни околини за поединечни проекти. Pipenv автоматски управува со зависностите и ги документира во Pipfile и Pipfile.lock, што гарантира дека проектите се репродуцибилни и стабилни. Оваа алатка е дизајнирана за да го олесни процесот на управување со зависности во Python проекти.

# **3 Детален опис на веб апликацијата**

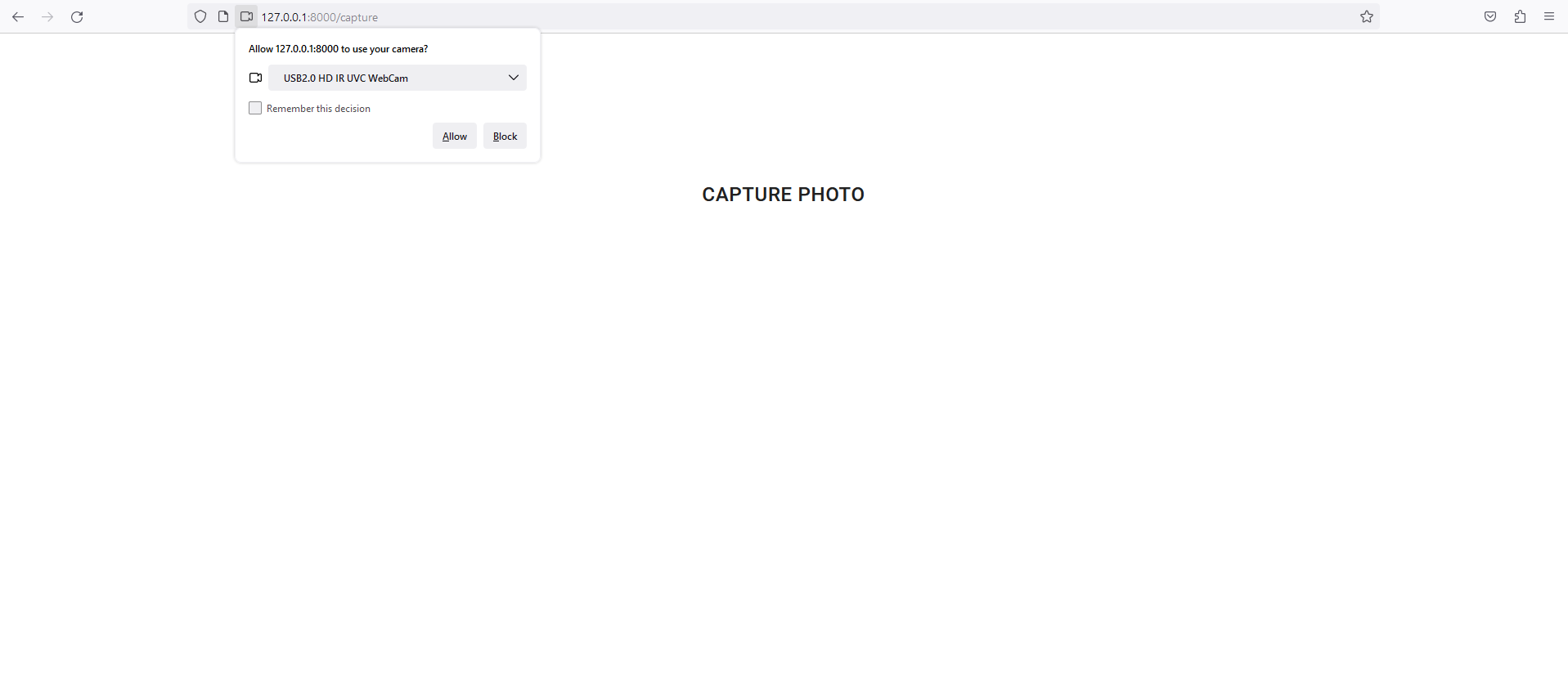
## 3.1 Почетна страна

При отварање на апликацијата се доведува до почетната страна на апликацијата која има 3 главни копчиња. Тие копчиња служат за внес на слика директно од камера или превземена слика од галерија, а пак 3-тото копче носи директно до страната за едитирање на почетна шаховска позиција каде корисникот ќе може мануелно да внесе било каква шаховска позиција и истата да се евалуира со Stockfish шаховскиот алгоритам.

****

Слика 1: Изглед на почетната страна

## 3.2 Страна за внесување на слика

****

Слика 2

По внесување на слика преку камера или од галерија, корисникот се пренасочува на **loading** страната каде внесената слика се испраќа до бекендот и се чека одговор од истиот. Ова се извршува преку следниов код:

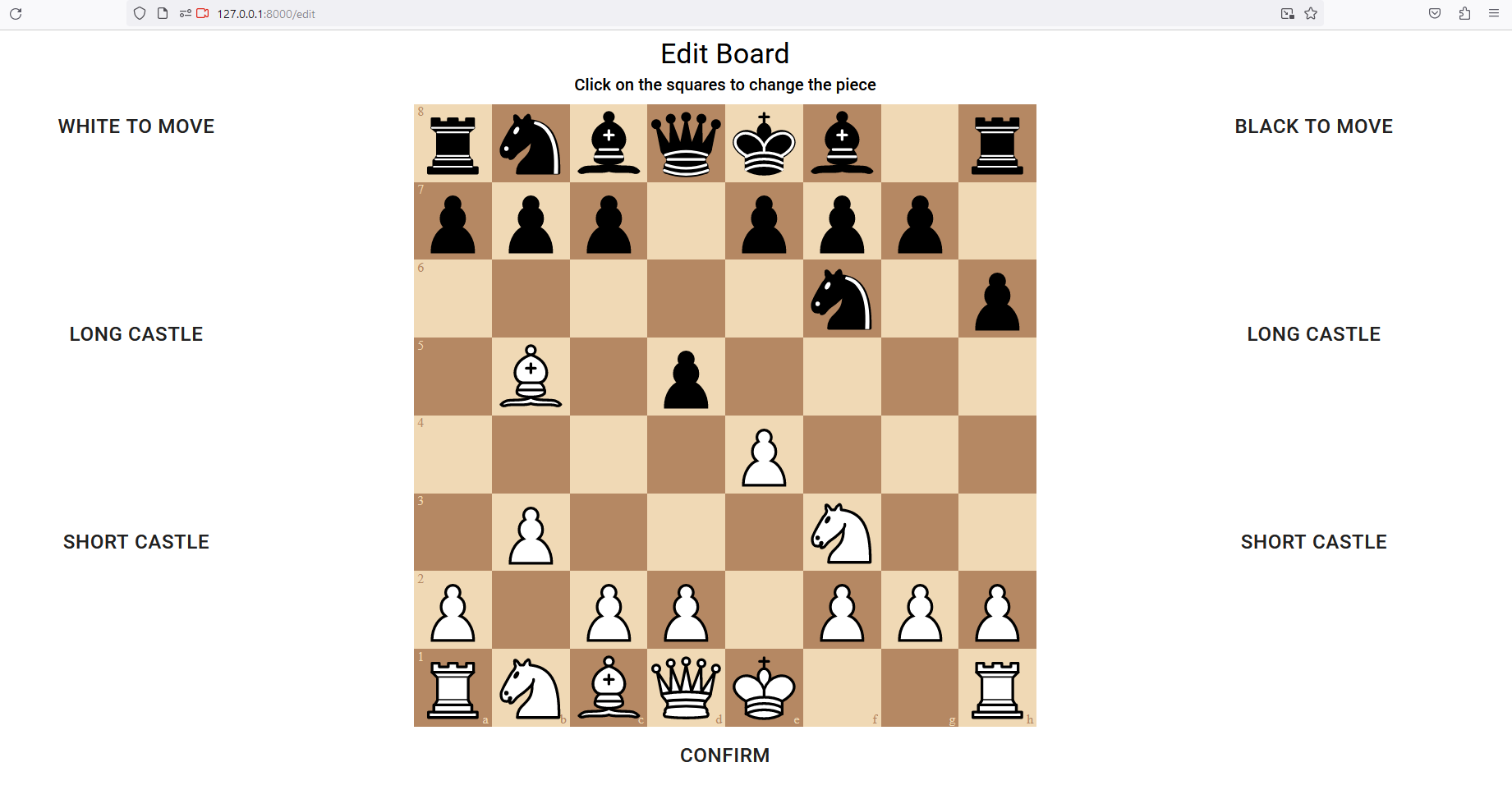


*Слика 3: Праќање на сликата до бекенд со помош на axios*

По завршување на процесирањето на сликата, бекендот испраќа одговор од шаховската позиција добиена од сликата во FEN формат така што корисникот автоматски се пренасочува до страната за едитирање на шаховската позиција.

## 3.3 Страна за промена на шаховската позиција

Во случај на погрешно детектирани шаховски фигури, корисникот може слободно да ги промени при клик на соодветниот квадрат така што при секој клик се менува шаховската фигура на полето во цикличен режим почнувајќи од бел пион па се до празен квадрат.

****

Слика 4: Изглед на страната за корекција на шаховската позиција

## 3.4 Главна страна

По внесување кој играч е на потег и опционален внес за кој играч има способност за рокада, по притаскање на копчето **confirm** корисникот се пренасочува на главната страна на веб апликацијата така што на корисничкиот интерфејс се прикажуваат трите најдобри потега од страна на играчот кој е на потег, така што стрелката обоена со темно зелена боја е најдобриот потег.

****

Слика 5: Изглед на главната страна

При секој потег се извршува следниов повик од фронтенд до бекенд и обратно.



*Слика 6: Кодот за преземање на 3те најдобри потега*



*Слика 7: Кодот за услужување на барањето од слика 6*

### 3.4.1 Функција за евалуација на шаховска позиција

Евалуација на шаховската позиција се остварува преку користење на еден од помоќните шаховски алгоритми наречен stockfish со верзија 16. Во рами на апликацијата се постава целосниот код потребен за stockfish да функционира, и со помош на пајтон библиотеката stockfish се интегрира во бекендот. За одредување на 3-те најдобри потега од дадена шаховска позиција, потребно е само да пратиме FEN.

A graph with a blue line

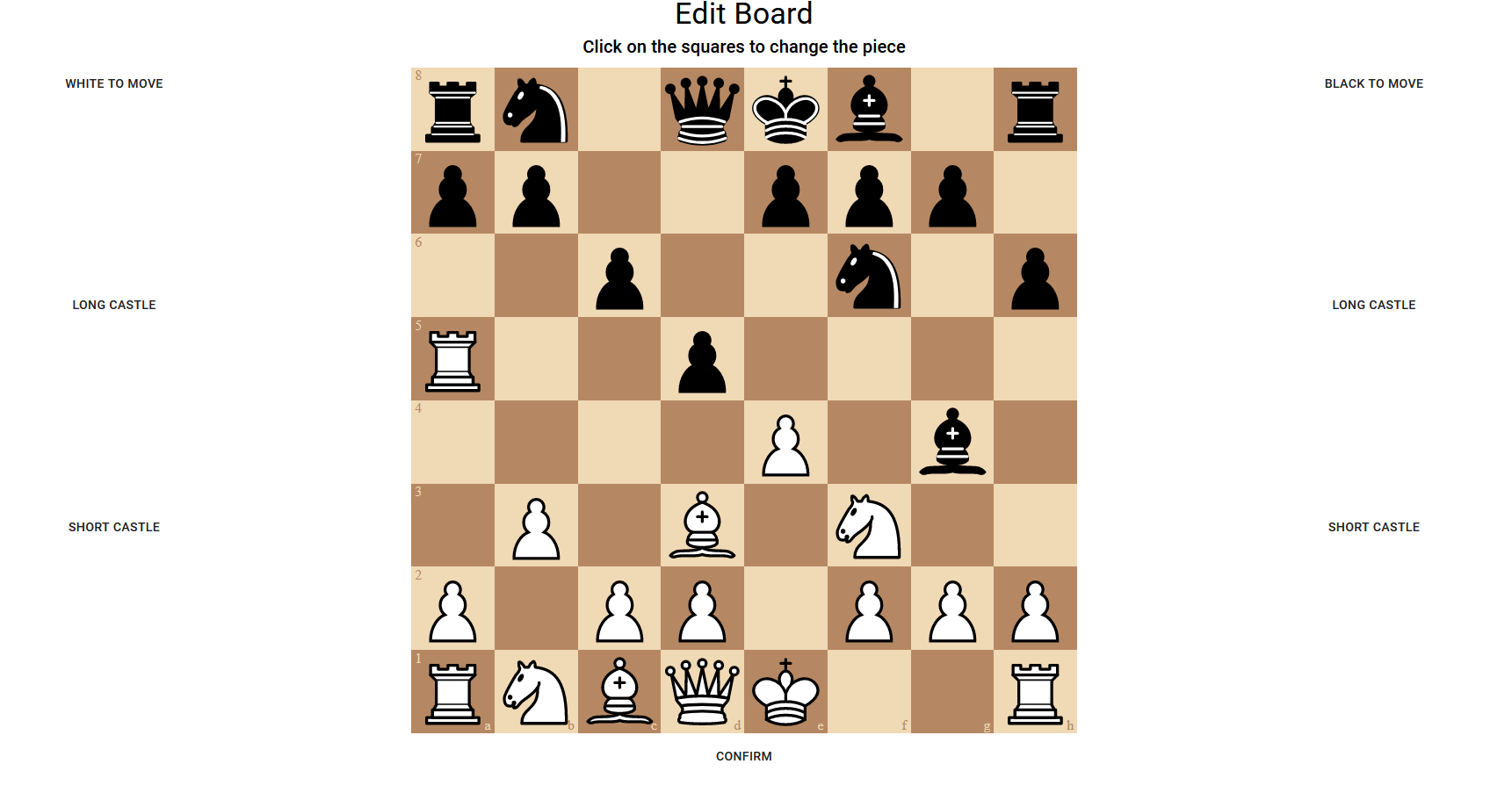
Description automatically generated

*Слика 8: Функција за*A graph with a green line

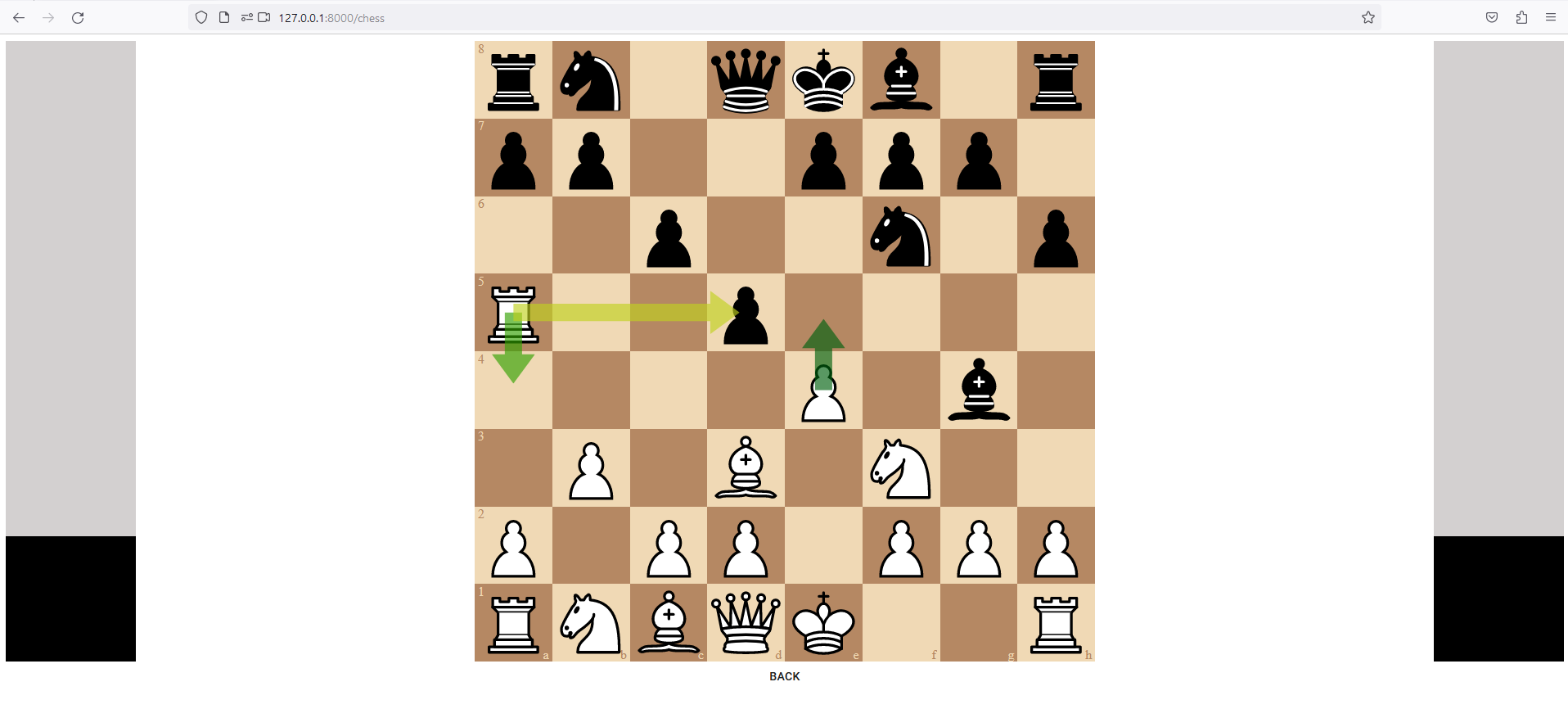
Description automatically generated

Слика 9:

## 3.5 Можност за навраќање

****

Слика 10: Додавање на произволна шаховска фигура (А5-Топ)

****

Слика 11:

Тука ставаш слики од дизајнот на страната, почетниот workflow diagram што го нацрта и дедално објаснување на компонентите напишани во реакт (ќе пополниш бајаѓи место) со слики се разбира

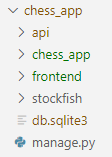
Почетна страна: при активирање на апликацијата се доведува до почетната страна на апликацијата која има 3 главни копчиња. Тие копчиња служат за внес на слика директно од камера или превземена слика од галерија, а пак 3тото копче носи директно до страната за едитирање на почетна шаховска позиција каде корисникот ќе може мануелно да внесе било каква шаховска позиција и истата да се евалуира.

А пак ако се прати слика, корисникот се носи на страна кое е како лоби (се чека хехе) да завршат сите потребни алгоритми за да вратам ФЕН стринг која ќе се искористи да сетира шаховската табла на страната за едитирање. Потоа корисникот конфигурира уште неколку копчиња за шаховската позиција како: дали двете страни имаат можност да направат рокада на секоја страна посебно и кој играч е на потег. При внес на овие информации и притискање на копчето, корисникот се носи на главната страна која опишува кој играч е во водство и кои се 3те најдобри потега според stockfish 16 шаховскиот алгоритам со длабочина од 15 од дрвото за пребарување на најдобар потег.

# **4 Структура на проектот**

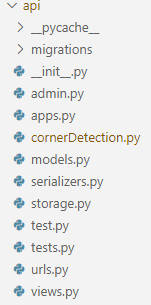
## 4.1 Главна структура

Целосната апликација е поставена во chess\_app папката, како во сликата.



* **API:** Целосниот код за бекендот на апликацијата.
* **chess\_app:** Иницијализација од Django Framework
* **Frontend**: Потребниот код за рендерирање на фронтендот со помош на React Framework
* **stockfish:** Потребен код за имплементација на stockfish шаховскиот алгоритам.

### 4.1.1 Бекенд



* **view.py:** Во овој фајл содржи две клучни функции: едниот е за послужување на барањето за предикција на шаховската табла и позициите на шаховските фигури, а пак другиот е послужување на барањето за превземање на најдобрите потега од дадената шаховска позиција.
* **models.py:** Вкупно има два модели, едниот модел е наменет за чување на сликата (ImageField) а пак другиот модел е за праќање на FEN формат (CharField)
* **cornerDetection.py:** Во овој фајл се чува целиот код потребен за препознавање на шаховската табла во дадена слика и предикција на четирите ќошеви на таблата, кодот за делење на сликата на 64 слики односно полиња што соодветствуваат на полињата на шаховската табла (А1, Б1 ...) потоа секоја поединечна слика се пушта на моделот за препознавање шаховски фигури и истите резултати се запишуваат на соодветната слика, односно шаховско поле
* **serializers.py:** Поставување на форматот на сликата за успешно разменување на сликата помеѓу бекендот и фронтендот.

### 4.1.2 Иницијализирање на Django

A screenshot of a computer

Description automatically generated

* **settings.py:** Поставување на клучни променливи како STATIC\_URL, MEDIA\_URL, MIDDLEWARE, INSTALLED\_APPS и други. Исто така се јави проблем каде при секое сликање на шаховската табла и праќање до бекендот (**view.py**) од поново се компајлираа моделите, затоа одлучив двата модели и stockfish шаховскиот алгоритам да ги иницијализирам кога започнува апликацијата, така што драстично се намали времето на процесирање на сликата.

### 4.1.3 Фронтенд

A screenshot of a computer

Description automatically generated

* **node\_modules:** Сите потребни модули за успешно рендерирање на фронтендот.
* **src:** Потребниот код за фронтендот.
* **templates:** Локација на главниот HTML код.
* **static:** Локација на оптимизираниот javascript код со помош на webpack и потребен статичен css код за подобрен кориснички интерфејс.
* **babel.config.json:** Kонфигурациски фајл кој овозможува успешно рендерирање на фронтендот во случај да се уклучи апликацијата преку постара верзија на веб прелистувач.
* **webpack.config.json**: Kонфигурациски фајл кој го оптимизира целосниот фронтенд така што ги поставува сите потребни ресурси (Javascript, css, слики итн.) за рендерирање на фронтендот во еден оптимизиран фајл.

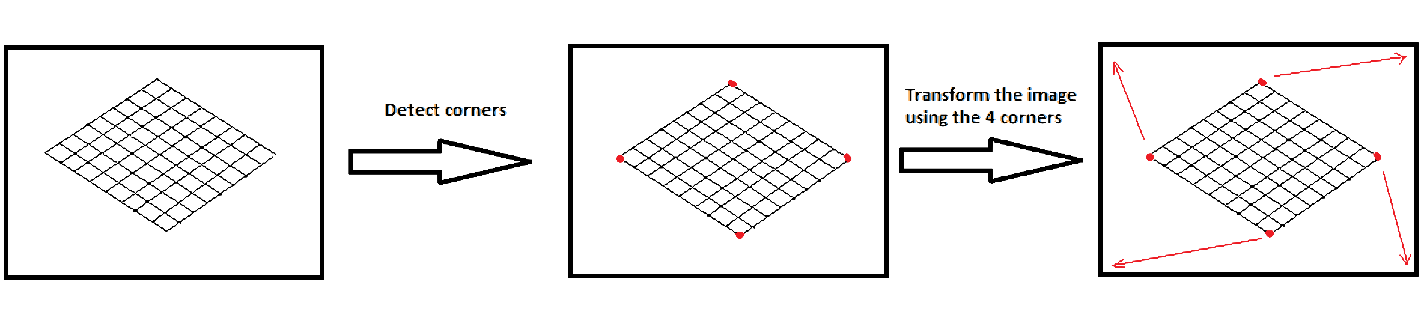
# **5 Модели за препознавање**

## 5.1 Модел за препознавање шаховска табла

За ова искористено е готов модел превземен од github страната, и со помош на пајтон библиотеката OpenCV прелагоден е моделот за зголемена ефикасност на детекција на шаховска табла.

Самата природа на алгоритмот доведува до зголемени преформаси ако аголот на сликата е од 90 степени поставена над шаховската табла. Бидејќи при делење на шаховската табла на 64 делови, се доведува до намалување на оверлап помеѓу полињата и со тоа се зголемува ефикасноста на моделот за препознавање фигури.

Дигитализација на физичката шаховска табла се извршува со помош на два модели. Првиот модел е задолжен за детекција на четирите ќошеви на шаховската табла, така што при детекција на 4 ќошеви се извршува низа на трансформации која доведува шаховската табла целосно да ја зафати сликата така што потоа со едноставно делење на сликата на 8 еднакви делови по x и y оската се добиваат 64 слики. Со првична претпоставка дека белиот играч се наоѓа на долната страна од сликата, можеме да ги лабелираме сите слики со соодветниот квадрат каде што се наоѓа почнувајќи од А1 квадратот од долната-лева страна на сликата.

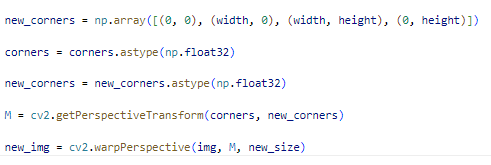


*Трансформација на сликата*

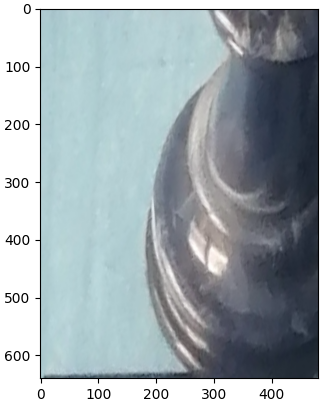
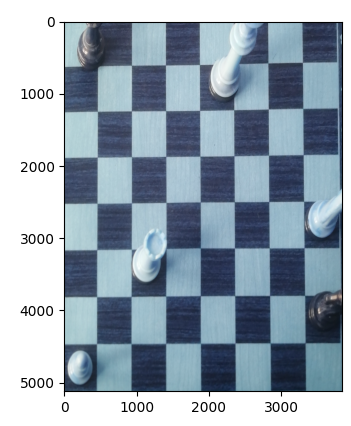
Пред да се детектираат кошевите на шаховската табла, мора сликата да се трансформира на должина 512 и ширина 384 за компатибилност со моделот. Но со тоа, координатите на детектираните кошеви на шаховската табла се одвиваат од опсег на квалитетот од трансформираната слика, но бидејќи не сакаме да се изгуби квалитетот на сликата се прави следнава трансформација на детектираните кошеви:



Со оваа трансформација се добиваат координатите на краевите на шаховската табла во релација со висината и ширината на оригиналната слика. Потоа се прави транформацијата на сл горе користејќи го следниов код со помош на OpenCV пајтон библиотеката:



Пример за горенаведената трансформација:



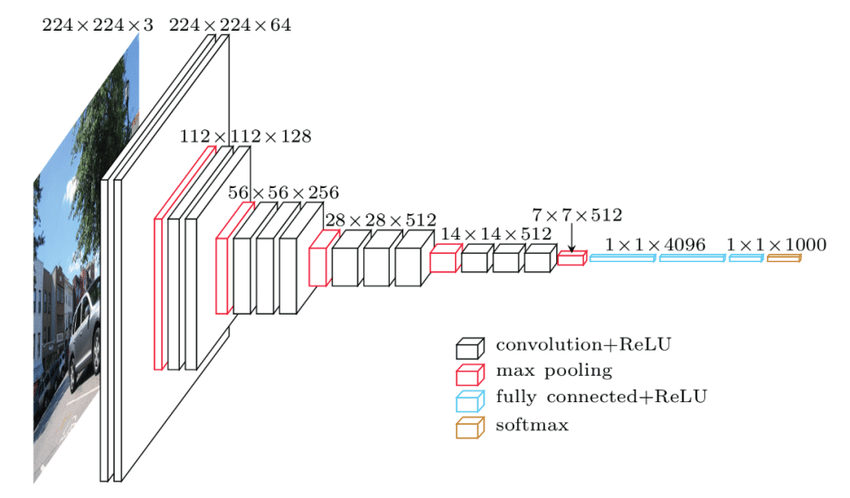
*Процесот на делење шаховската табла на 64 слики*

По добивање на координатите на сите 64 полиња од шаховската табла, истите можат да се искористат со трансформацијата за да се добие низа од 64 слики кои кореспондираат со полето од шаховската табла. Со ова лесно може да се добие FEN стринг-от, така што ќе биде пратен како одговор на фронтендот.

## 5.2 Модел за препознавање шаховски фигури

### 5.2.1 Готов модел (VGG16)

VGG16 е модел за конволуциска невронска мрежа која била разработена и истренирана од страна на Симонјан и Зисерман во 2014 година. Оваа мрежа е тренирана на ImageNet, сет на податоци кој содржи повеќе од 15 милиони слики. VGG16 се состои од 13 конволуциски слоеви, 5 max-pooling слоеви и 3 целосно поврзани слоеви. Во почетните слоеви, мрежата ги детектира нискорангирани карактеристики како мали ивици или зони со боја. Овие карактеристики се комбинираат за да се формираат високорангирани карактеристики во понапредните слоеви на мрежата. Со овие високорангирани карактеристики се одлучува класификацијата на влезната слика, односно излезот на моделот.



*Архитектура на VGG16*

Во рамките на нашиот модел, користиме процес познат како "пренос на учење" (transfer learning). Преносот на учење во конволуциски невронски мрежи (CNN) вклучува користење на модел кој е предходно трениран на голем сет на податоци и пренесување на неговото знаење кон помал сет на податоци. Овој модел ќе биде искористен како подлога за тренирање на моделот за препознавање шаховски фигури, така што ќе биде отстранет крајнот слој на моделот и на негово место ќе се додаде нов слој кој ќе биде трениран со слики од шаховски фигури.

### 5.2.2 Податочно множество

Бидејќи секој шаховски сет е уникатен морав да изградам сопствено податочното множество од мојот шаховски сет, така што сликав неколку конфигурации на шаховската табла со фигурите и ги сечев поединечно секој квадрат од шаховската табла.

Лабелирање на сликите беше изведено така што сликите за тренирање се поделени во 13 папки именувани според секоја уникатна фигура плус празното поле. Пример за бела кралица, папката е именувана **wq** (white queen) и сликите соодветно се поставени во своите папки. Со ова, подготвени се податоците за тренирање на моделот.

### 5.2.3 Структура на мрежата

Додадени се вкупно 3 слоеви на VGG16 моделот од кој 2 се конволуциски (Conv2D) со ‘ReLu’ активациска функција и последниот слој кој има 13 излези. Исто така додадено е dropout слој за намалување на шансите за overfitting.

A screenshot of a computer program

Description automatically generated

*Архитектура на додадениот слој*

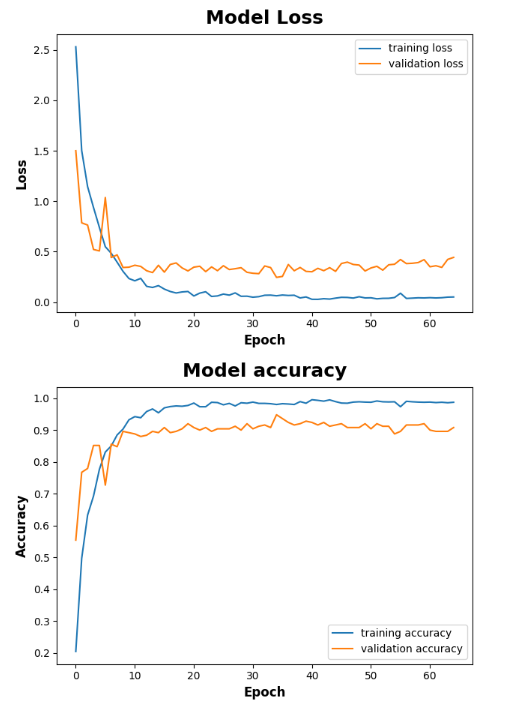
### 5.2.4 Тренирање на моделот

За тренирање на моделот искористено се следниве хипер-параметри:

* **Adam** оптимизатор за оптимизација на тежините на моделот.
* **categorical\_crossentropy** како функцијата на загуба (loss function).
* **categorical\_accuracy** како метрика за прецизност на класификацијата.

Исто така додадено е метод за порано запирање на тренирањето на моделот откако нема да има значително подобрување на метриката за прецизност после 15 епохи, со ова се справуваме со overfitting на моделот.

### 5.2.5 Резултати од тренирање на моделот

A screenshot of a computer

Description automatically generatedA graph with different colored squares

Description automatically generated

# **6 Можности за проширување на веб апликацијата**

Во рамки на целосната апликација има можности за проширување во секој аспект по апликацијата. Како една од поинтересните делови за проширување на апликацијата но и значително потежок за имплементација е додавање начин корисниците да можат да внесат одреден број на слики од нивната уникатна шаховска табла со шаховските фигури, која при процес наречен transfer learning ќе го надгради моделот со спесифичните шаховски фигури и ќе ја зголеми прецизноста на моделот за специфичната форма на шаховските фигури. Секако со оваа функционалност би било конвенционално да се направи функционалност за најавување на корисниците за можност на зачувување на уникатниот модел.

Друг аспект на проширување на апликацијата е додавање имиња и објаснувања на познати шаховски отварања. Исто така би било корисно и интересно, во случај на одредена шаховска позиција да има секвенција на шаховски потези која доведува до форсиран шах мат, истите потега да се прикажат на корисникот.

Додавање на функционалност во случај кралот да не е на стартната позиција, копчињата за рокада од соодветната страна да бидат автоматски селектирани.

# **7 Заклучок**

Линк до Github

<https://github.com/BojanR224/django_react_app>

Линк до Google Collaboratory

[https://colab.research.google.com/drive/1cDMCZmawemunPZf6B5YqEYYFUO0-mE2w#scrollTo=ywq7wkgmUJtk](https://colab.research.google.com/drive/1cDMCZmawemunPZf6B5YqEYYFUO0-mE2w%23scrollTo=ywq7wkgmUJtk)

**Литература**

1. React, 2023. URL <https://react.dev/>

2. Django, 2023. URL <https://docs.djangoproject.com/en/4.2/>

3. Pipenv, 2023. URL <https://pipenv.pypa.io/en/latest/>

4. Stockfish version 16, 2023. URL <https://stockfishchess.org/>

5. VGG16 Model, 2014. URL <https://www.mathworks.com/help/deeplearning/ref/vgg16.html>