**Универзитет у Београду**

**Факултет организационих наука**

**Пројектни рад**

**Sudoku ++**

**Алати и методе вештачке интелигенције и софтверског инжињерства**

Ментор: Студент:   
Др Драган О. Ђурић Јелена Репац 2023/3811

Београд, 2024.

Садржај

[1 Вербални опис 3](#_Toc158721095)

[2 Корисничко упуство 4](#_Toc158721096)

[3 Имплементација 9](#_Toc158721097)

[3.1 Алгоритми 11](#_Toc158721098)

[3.2 Руте 14](#_Toc158721099)

[4 Закључак 18](#_Toc158721100)

# 1 Вербални опис

Sudoku++ је апликација која пружа кориснику интерактивно и забавно искуство решавања судокуа. Осим што нуди различите нивое тежине судоку табли, апликација генерише потпуно насумичну таблу за сваку игру.

У процесу решавања, корисници могу користити помоћ која траје 20 секунди, током којих им се приказују могуће вредности за оно поље на којем се тренутно налазе. Додатно, имплементирана је валидација за унос појединачног броја на одређену позицију. У случају да број није одговарајућ, узимајући у обзир правила Судоку игре, број постаје црвен, упозоравајући играча да промени унети број. Игра се покреће кликом на таблу, након чега почиње и одбројавање времена, које се након успешног решавања чува као резултат тог играча.

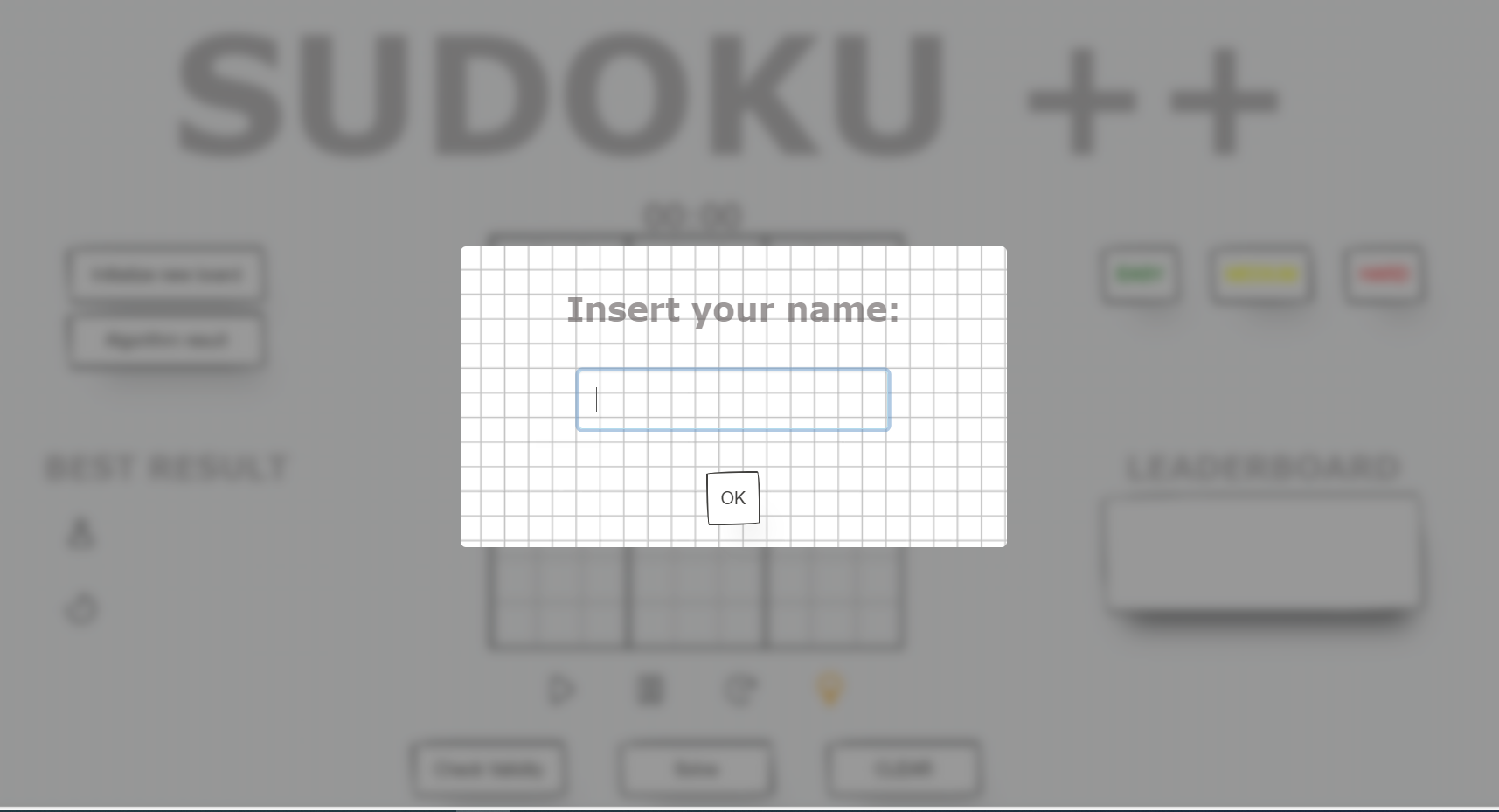
Као додатна функционалност, могуће је добити информацију о томе колико је алгоритму потребно да би решио судоку.

У сваком тренутку, играч може да добије решење целе табле кликом на дугме *Solve*.

Да би се омогућило праћење резултата и подстакао такмичарски дух, приликом покретања апликације неопходно је унети ваше име. Што вам омогућава да пратите свој напредак и да га поредите са резултатима других играча.

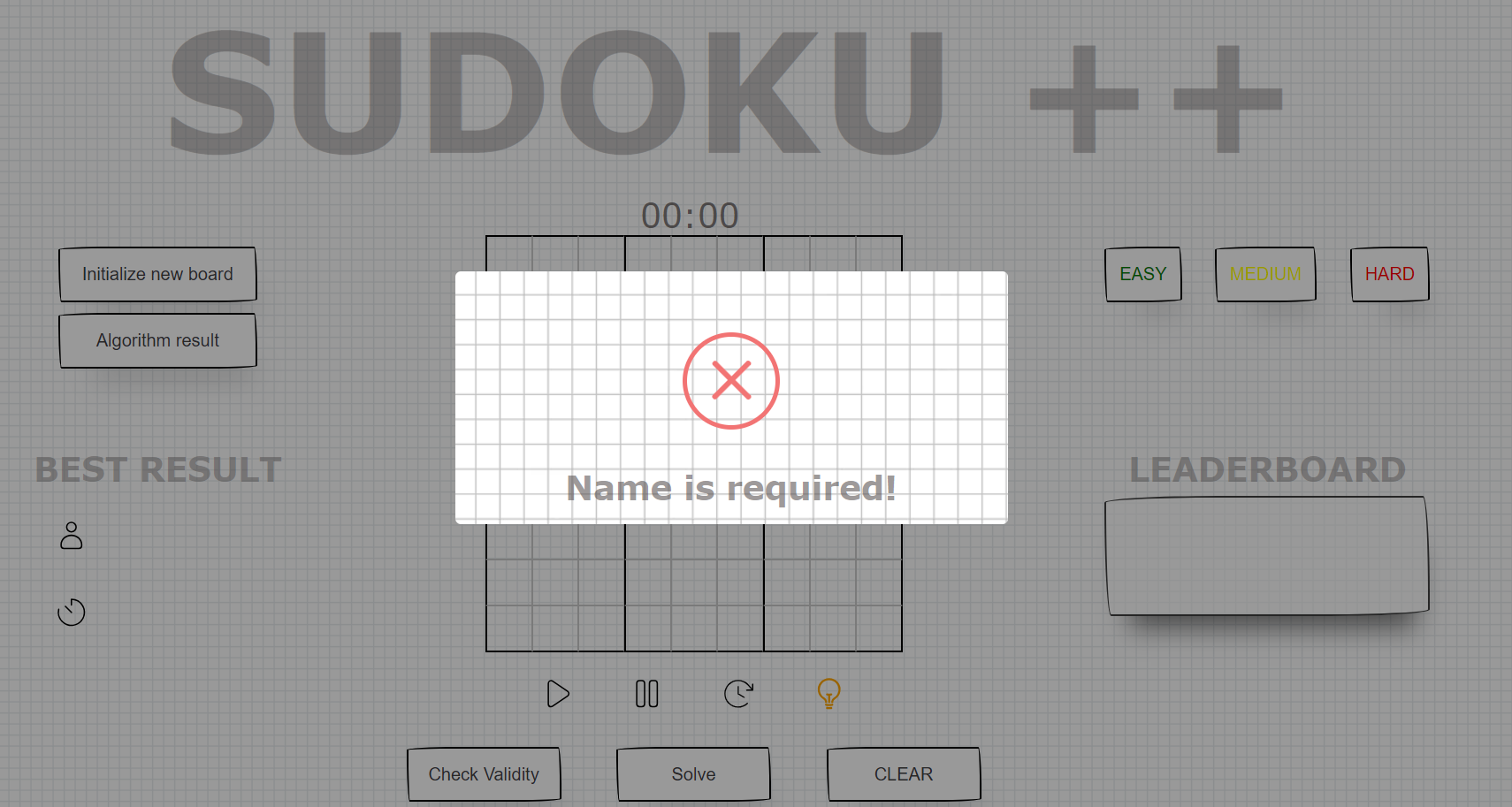
# 2 Корисничко упутство

Изглед почетног екрана је дат на слици испод. Неопходно је да корисник унесе своје име како би могао да започне игру.



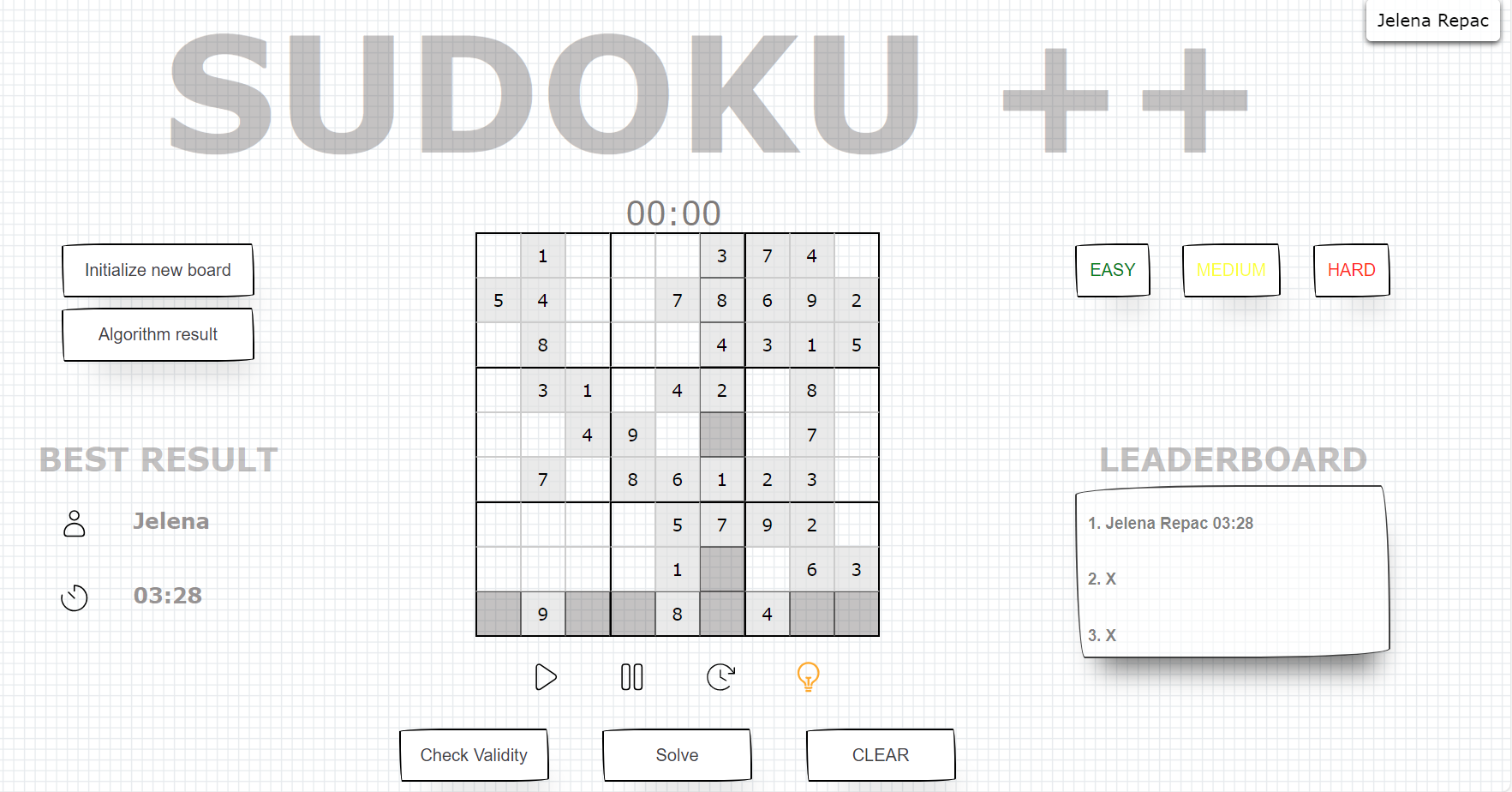
Слика 1 Почетни екран - унос имена

У случају да се име не унесе, добија се следећа порука.



Слика 2 Порука - Неопходно је унети име

Након уноса имена, кориснику се приказује изгенерисана судоку табла и он може отпочети своју игру. Почетна табла је изгенерисана уз помоћ алгоритма који је описан у следећем поглављу.

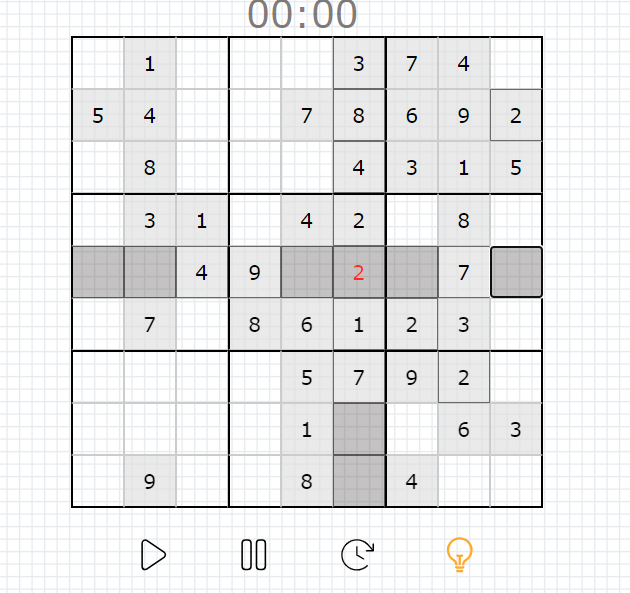


Слика 3 Почетна страна - након пријаве

У горњем десном углу приказан је тренутно активни корисник.

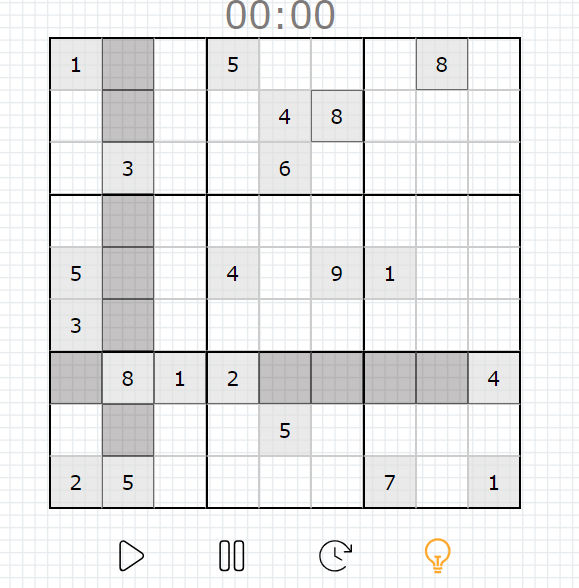
Са десне стране постоји опција за генерисање судокуа различите тежине, као и приказ прва три најбоља резултата. Док је са леве стране приказ тренутно најбољег резултата.

Како би се време стартовало, играч мора да крене да решава судоку. Валидација је присутна за сваки унос, а у случају да број није одговарајући на тој конкретној позицији, постаје црвен и на тај начин обавештава корисника да би требало да промени унет број.



Слика 4 Неисправно унет број

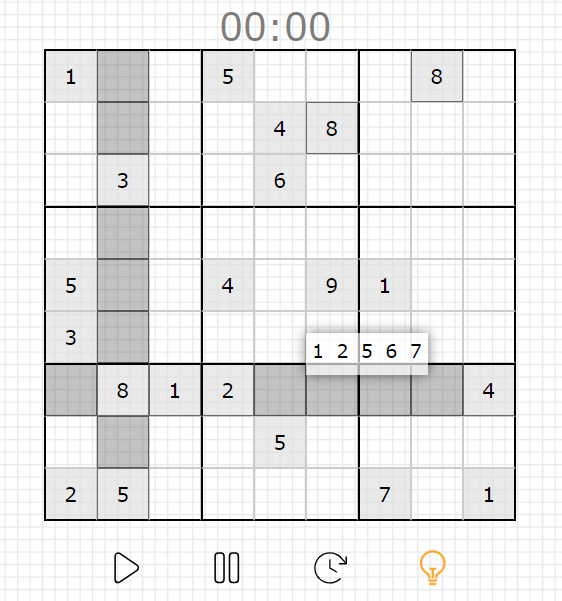
Такође, кликом на одређени број, маркира се ред и колона у којој се број налази, као и то да се означе сва она поља која садрже исти тај број, како би сигнализирали кориснику на којим местима не би требало да се упише.



Слика 5 Маркирање недозвољених ћелија за селектован број

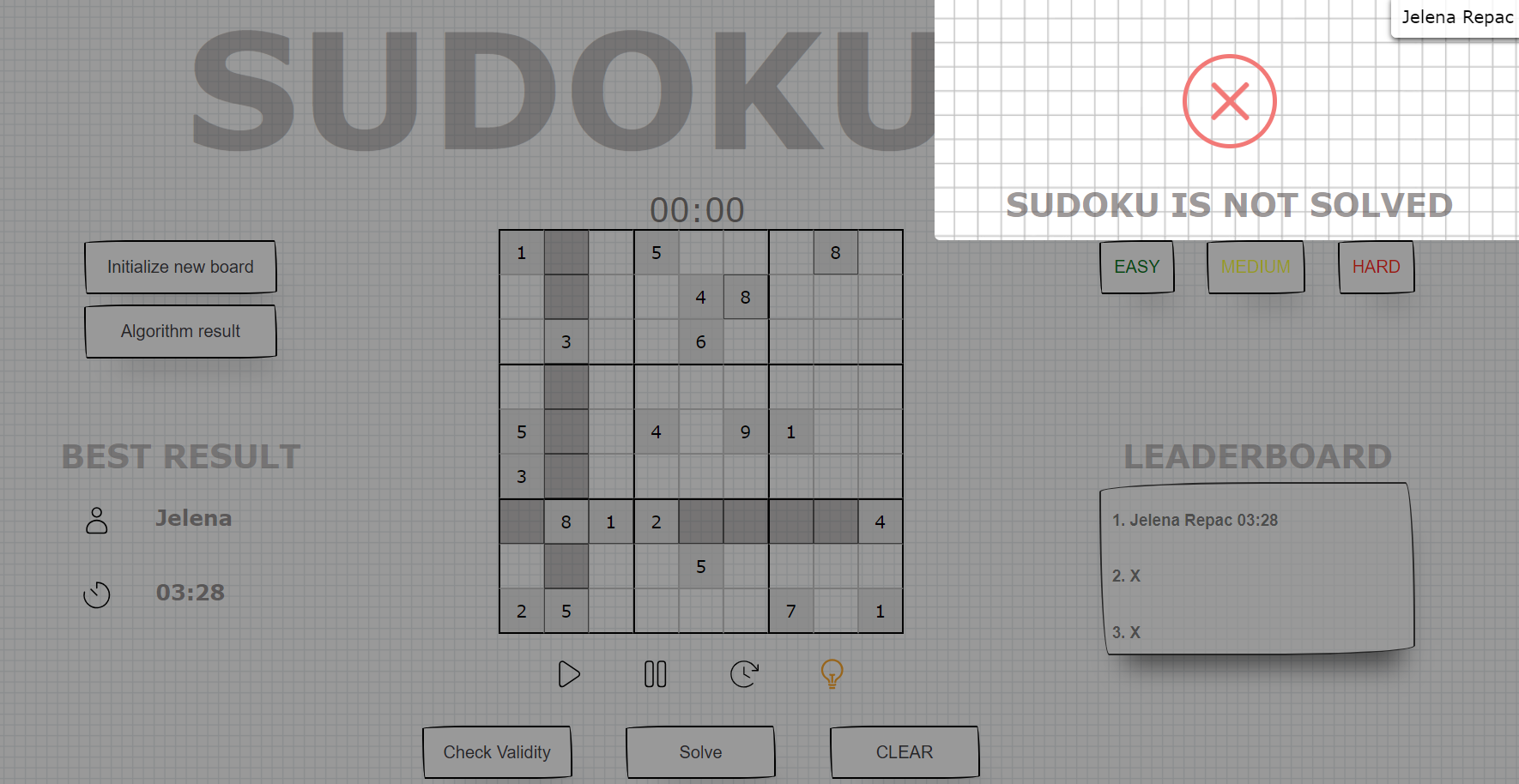
Време може да се заузстави, паузира и рестартује на дугмиће који се налазе испод табле.

Корисник има право на помоћ од 20 секунди. Кликом на лампицу приказују му се могуће вредности за свако поље на којем се налази.



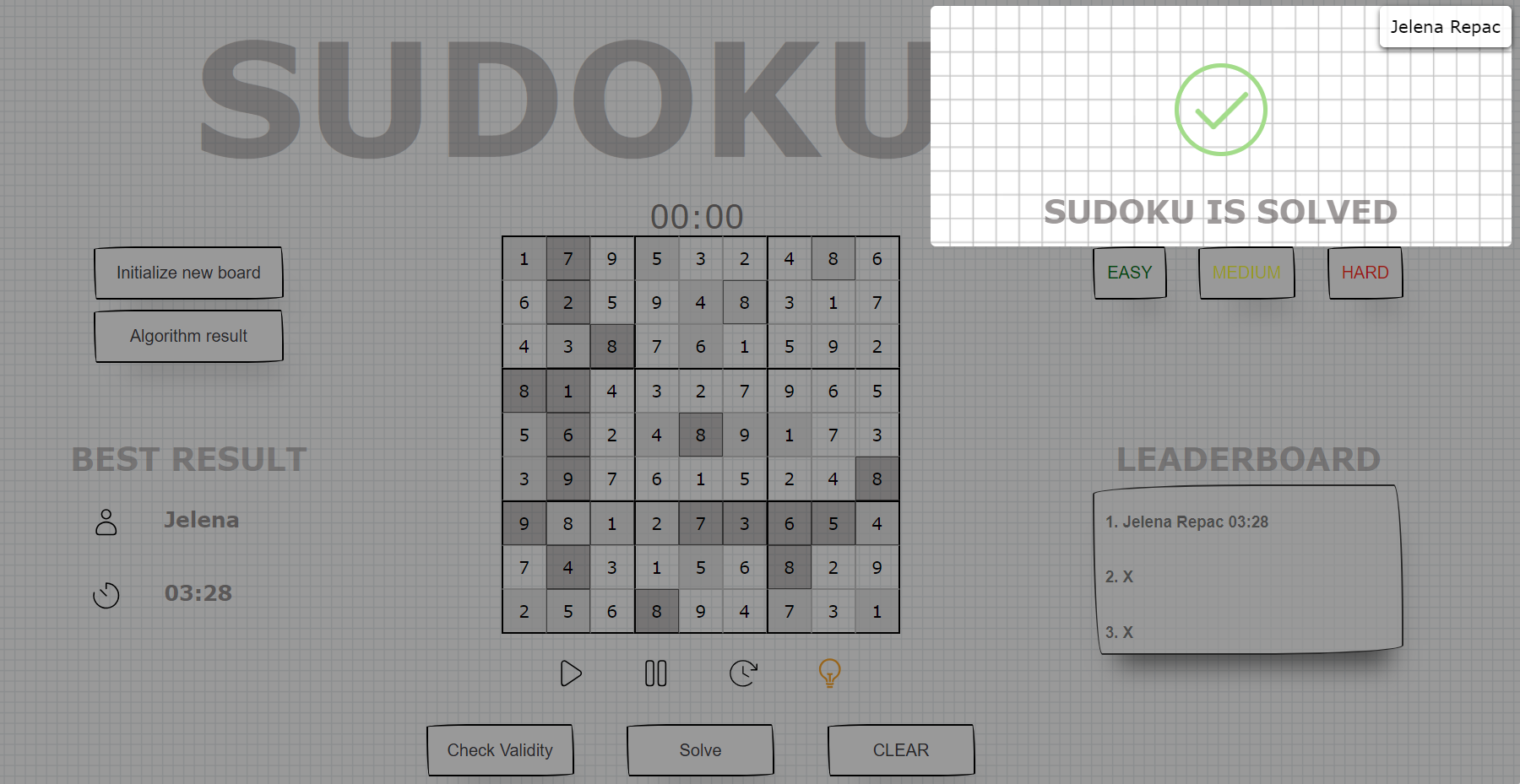
Слика 6 Помоћ у виду могућих вредности

Кликом на дугме *Check validity* корисник проверава исправност судоку табле. У случају да судоку није решен, приказује се следећа порука.



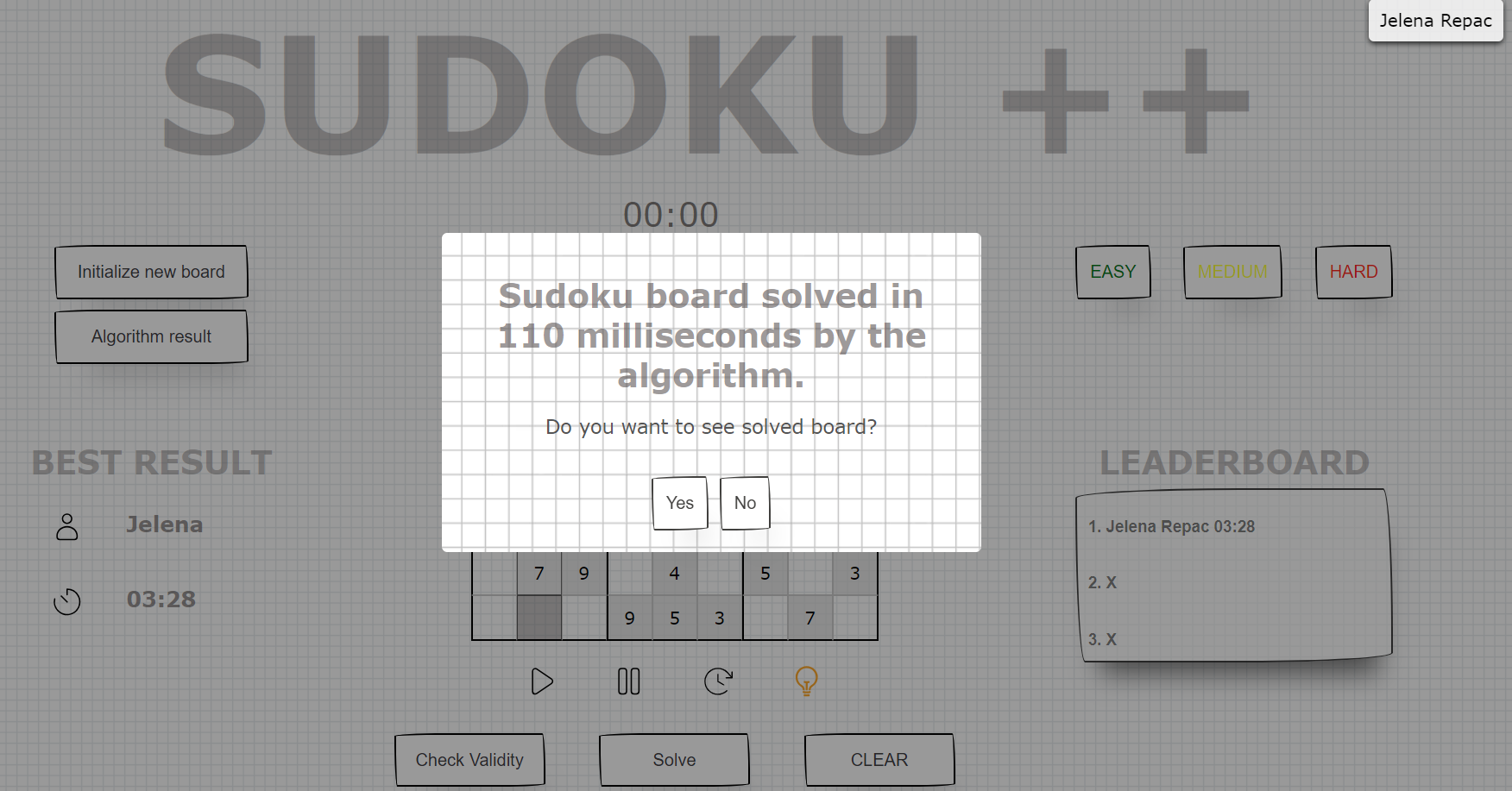
Слика 7 Порука о грешци - Судоку није решен

У супротном, кориснику се приказује следећи *pop up.*



Слика 8 Порука о успешности - Судоку је решен

Такође, корисник може видети колико је времена потребно алгоритму да реши задати судоку, кликом на дугме *Algorithm result.*



Слика 9 Времен потребно за решавање судокуа уз помоћ алгоритма

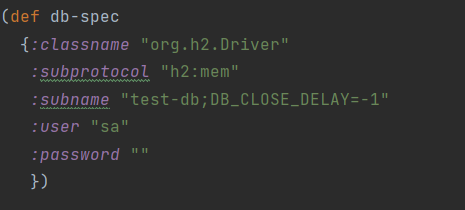
# 3 Имплементација

У апликацији је имплементирана *H2 embedded* база података, која је дистрибуирана као библиотека. Подаци се чувају у .mv.db датотекама, док се сами SQL упити извршавају у Clojure-u. Оваква база података омогућава развој и тестирање апликације са интегрисаном базом података без потребе за спољним сервером.

За саму имплементацију базе, било је неопходно укључити следећи *dependency*:

[com.h2database/h2 "1.4.200"]

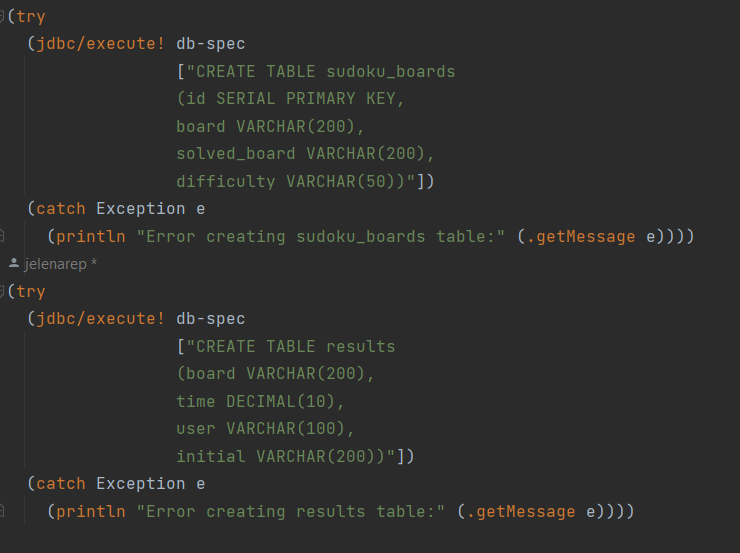
Док је сама спецификација базе дефинисана на следећи начин.



Слика 10 Спецификација базе података

DB\_CLOSE\_DELAY је сетован на -1, како се веза са базом не би аутоматски затворила, односно како бисмо задржали базу активну у меморији током трајања рада апликације.

Првобитно су креиране две табеле, *SUDOKU\_BOARDS* и *RESULTS.*

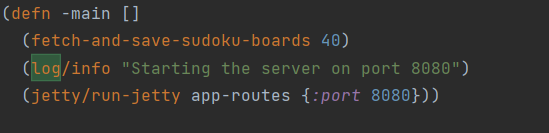


Слика 11 Креирање табела

*SUDOKU\_BOARDS* чува судоку табле инсертоване позивањем јавног API-ja <https://sudoku-api.vercel.app/api/dosuku>. Док табела results чува резултат (време за које је судоку решен) као и конкретне user-e који су играли судоку.

[ring/ring-jetty-adapter "1.6.3"] је библиотека која омогућава покретање веб сервера помоћу Jetty сервера. Ring представља стандардни интерфејс за комуникацију између Clojure aпликација и HTTP сервера.

Сервер се покреће уз помоћ jetty/run-jetty

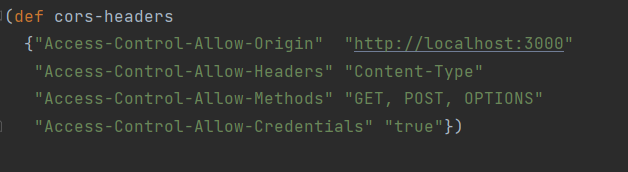


Слика 12 Мain функција за покретање апликације

Проласком кроз *app-routes* дефинисаних помоћу библиотеке [compojure "1.7.1"], омогућавамо рутирање унутар наше апликације.

Додата је и библиотека за логовање [org.clojure/tools.logging "1.2.4"].

Како би омогућили комуникацију са фронт делом апликације неопходно је додати заглавље у *request*. На овај начин је омогућен CORS (*Cross Origin Resource Sharing*) у оквиру веб апликације. Док је приступ ресурсима на страни Clojure апликације омогућен захтевима са адресе <http://localhost:3000>**.**

. 

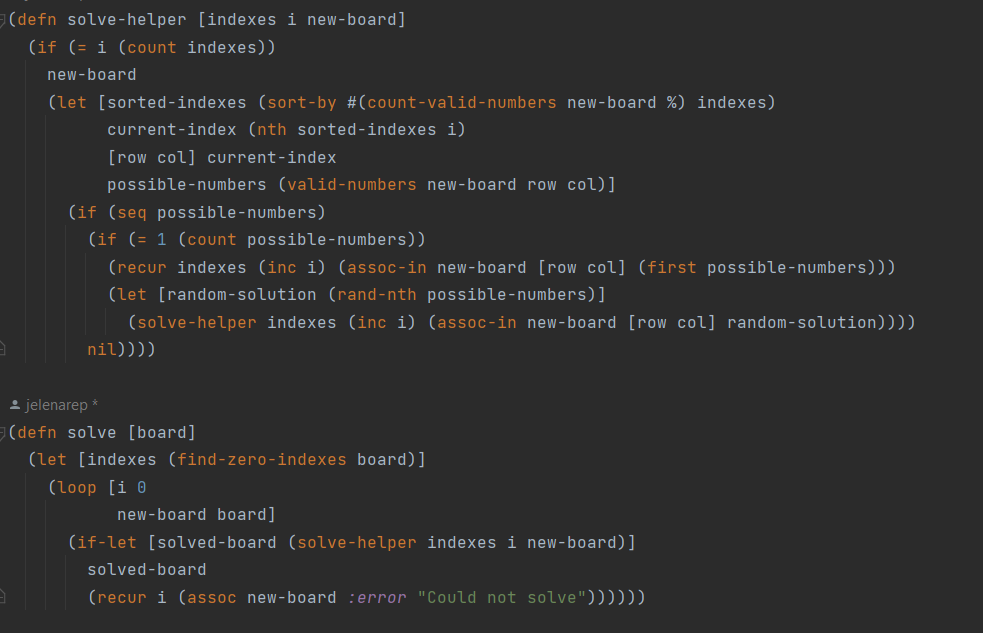
Слика 13 Заглавље

Покретањем **sudokiApi.clj,** покрећемо целу апликацију и излажемо endpoint-e који омогућавају интеракцију са апликацијом.

## 3.1 Алгоритми

Решавање судоку табли је имплементирано на следећи начин:

* *easy* - решава се уз помоћ алгоритма у наставку



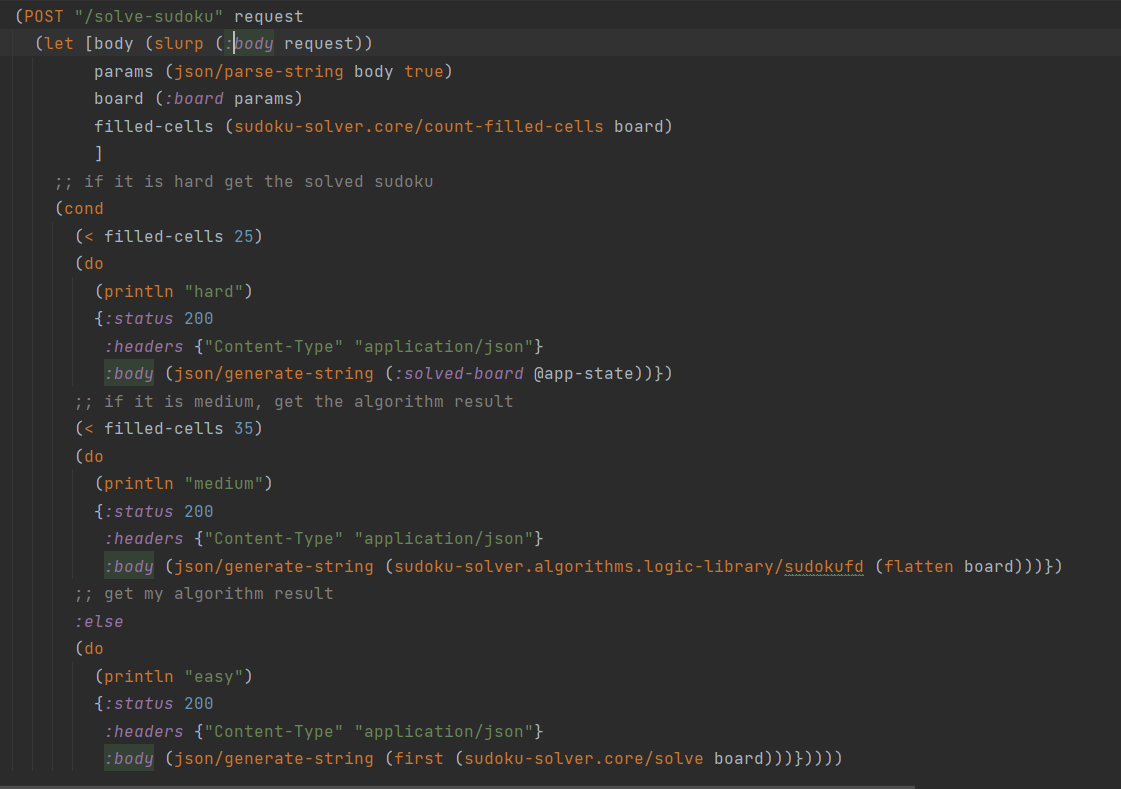
Слика 14 Функција за решавање судокуа

Алгоритам је заснован на проналаску оних позиција са само једним решењем, уколико постоји више могућих бројева за поље, бирамо један насумичан број и рекурзивно настављамо да тражимо решење.

* *medium* - позива се алгоритам који је преузет са <https://github.com/sideshowcoder/core-logic-sudoku-solver/tree/master>
* *hard* - решен судоку са позваног api-ja

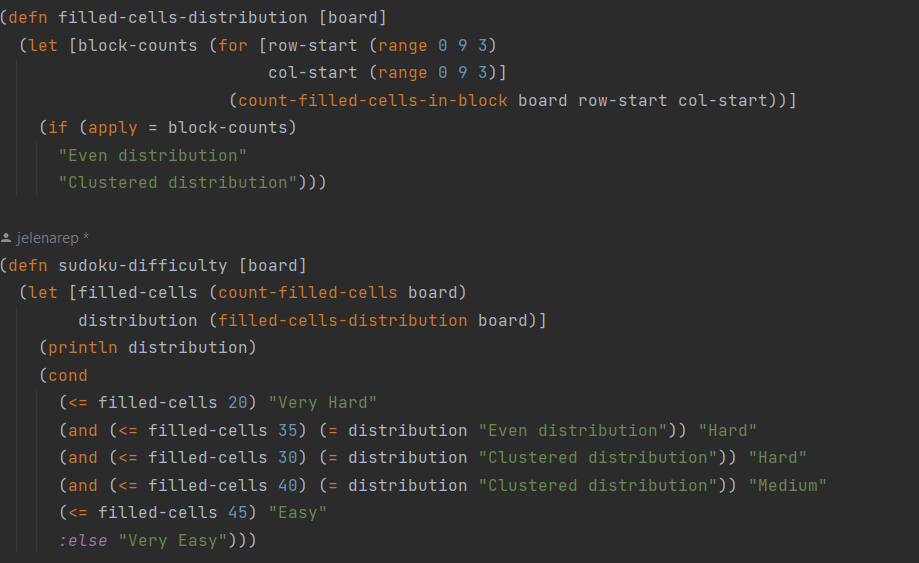
<https://sudoku-api.vercel.app/api/dosuku>.

Описан процес је приказан на следећој слици.



Слика 15 Процес решавања

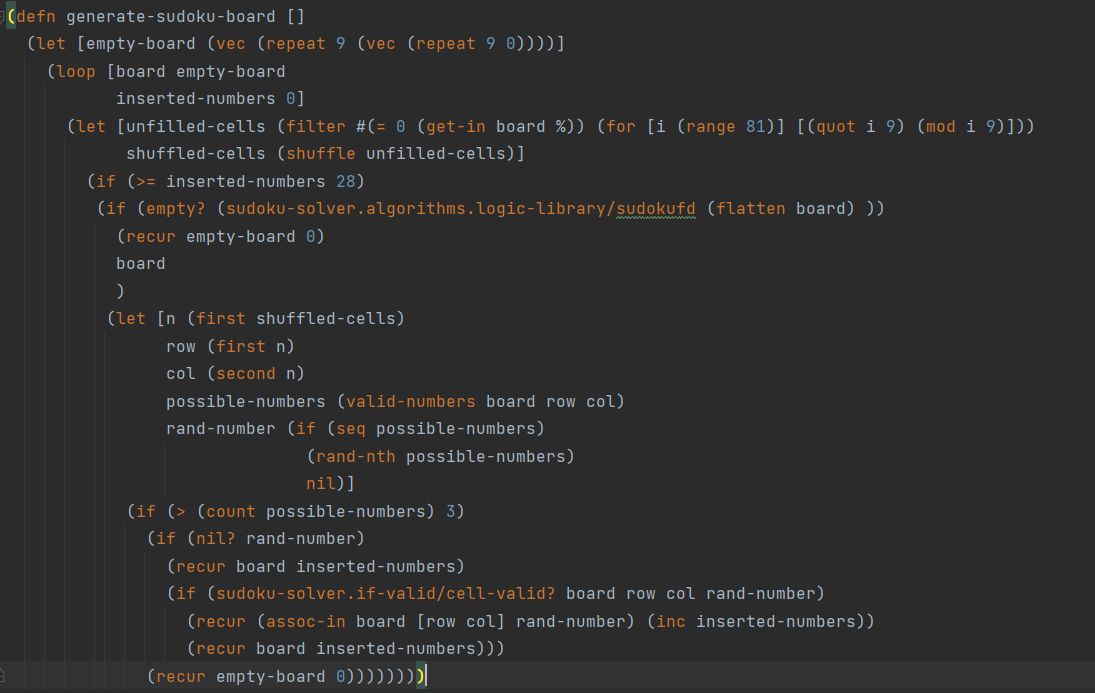
У случају решавања, тежину смо поделили на три нивоа, у зависности од броја попуњених ћелија (*easy, medium, hard*).  
Док смо за генерисање нових табли користили функцију која одређују тежину табле на основу броја попуњених ћелија и на основу распореда истих. На овај начин нудимо кориснику прецизније одређену тежину насумично генерисане табле.



Слика 16 Процена тежине судоку табле

Нову таблу можемо добити на следеће начине:

* Преузимање табли са јавног API-ja
* Позивањем функције у наставку за генерисање судоку табле



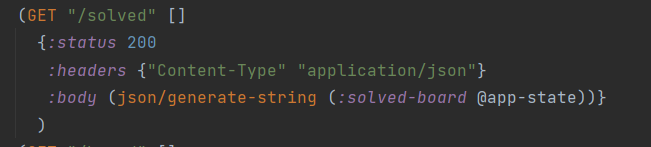
Слика 17 Генерисање судоку табле

Иницијално се креира празна табла 9x9, са свим елементима постављеним на 0. Кроз алгоритам се прати број оних ћелија које нису попуњене, а на основу њих се генерише случајан распоред празних ћелија. Како бисмо били сигурни да је изгенерисана табла решива позива се преузет алгоритам који проверава да ли је могуће решити судоку. Што значи да уколико табла није валидна, односно нема решење, враћамо се на корак један и алгоритам почиње генерисање нове табле. Како би се табла попунила, узима се прва ћелија из листе и уколико постоји више од 3 могућа броја за ћелију, бира се један насумично, у супротном, алгоритам се враћа на почетак. Још један обавезан корак, позивање функције **cell-valid?,** која проверава ли је изабрани број валидан за ту конкретну ћелију узимајући у обзир целокупну таблу. Целокупан поступак се понавља све док не буде 28 попуњених ћелија.

## 3.2 Руте

1. GET /solved

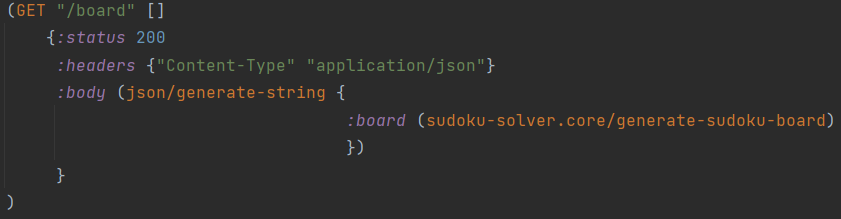
Слање решене табле која се чита из атома у који је смештена приликом позивања јавног API-ja



Слика 18 GET /solved

1. GET /board

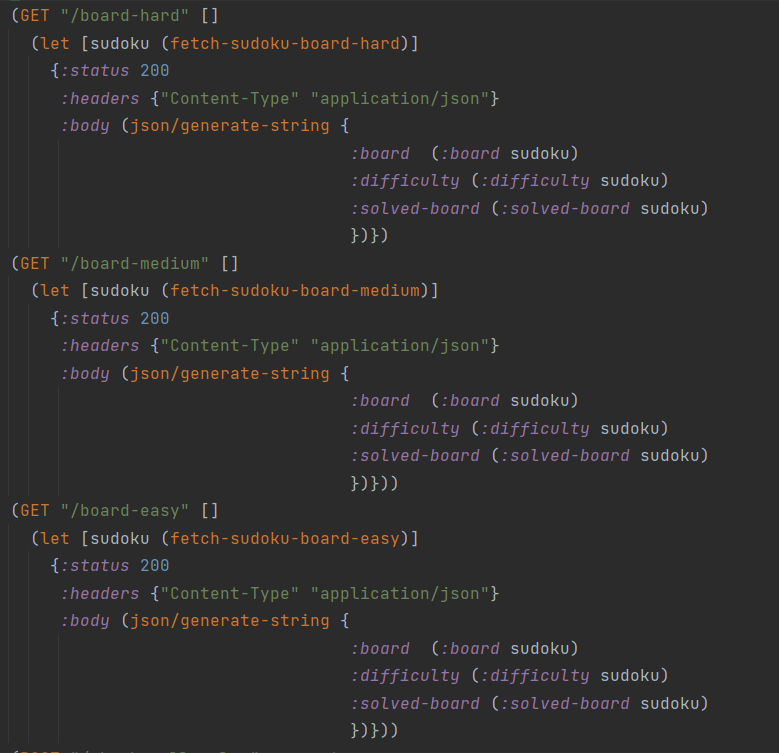
Слање изгенерисане табле



Слика 19 GET /board

1. GET /board-easy /board-medium /board-hard

У оквиру следеће три руте, врши се позивање базе како би се издвојила табла конкретне тежине



Слика 20 GET /board-easy

1. POST /check-cell-value

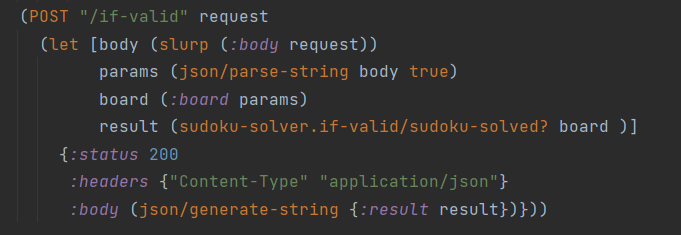
Провера да ли је број одговарајућ на унетој позицији



Слика 21 POST /check-cell-value

1. POST /if-valid

Провера да ли је табла решена



Слика 22 POST /if-valid

1. POST /cell-hints

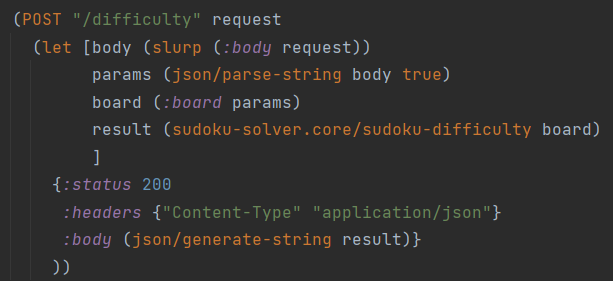
Добија се листа могућих бројева за конкретно поље



Слика 23 POST /cell-hints

1. POST /difficulty

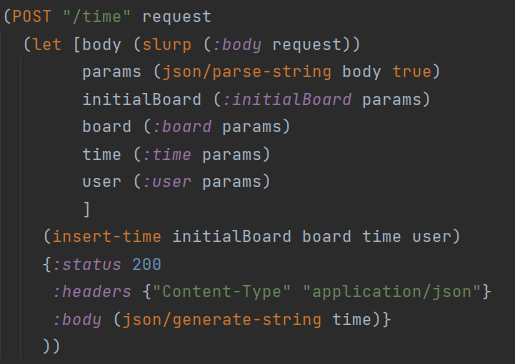
Враћа која је тежина прослеђене табле



Слика 24 POST /difficulty

1. POST /time

Уписује се време које је било потребно кориснику да реши судоку



Слика 25 POST /time

1. GET /best-result

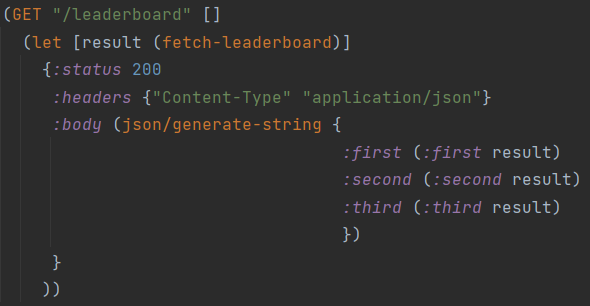
Добија се најбољи резултат решавања



Слика 26 GET /best-result

1. GET /leaderboard

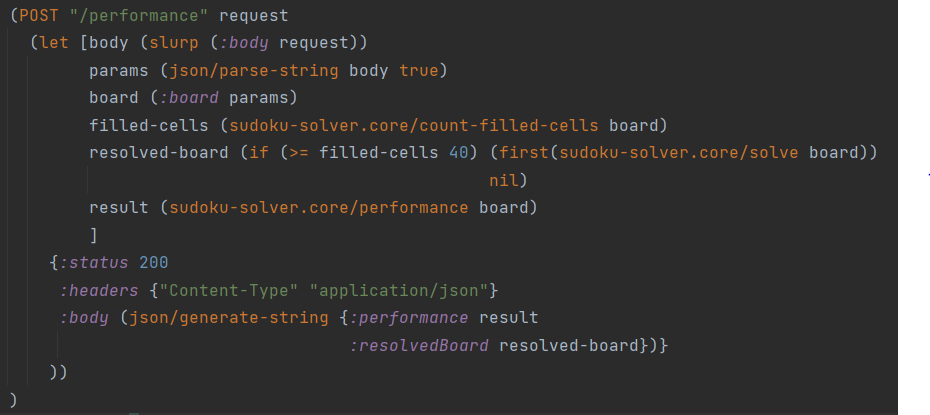
Добијају се прва три најбоља резултата



Слика 27 GET /leaderboard

1. POST /performance

Добија се време које је потребно за решавање табле уз помоћ алгоритма.



Слика 28 POST /performance

# 4 Закључак

Sudoku ++ је апликација која комбинује забаву, изазов и могућност праћења напретка корисника. Развијана је као *full stack* апликација, уз помоћ ReactJS технологије за кориснички интерфејс и Clojure-a за развој целокупне логике која је неопходна за једну Судоку игру.