SPRAWOZDANIE Z REALIZACJI PROJEKTU INDYWIDUALNEGO

Wykonała: Agata Lachowiecka
Sprawdzający: mgr inż. Paweł Zawadzki
Data: Czerwiec 2021

Spis treści

1 Wstęp

1.1 Cel dokumentu

Celem dokumentu jest podsumowanie wykonanego projektu realizowanego w ramach przedmiotu Projekt Indywidualny, przedstawienie efektów końcowych oraz ocena wykonanej aplikacji i pracy nad nią.

1.2 Cel projektu

Celem projektu było stworzenie aplikacji desktopowej będącej grą w przechodzenie labiryntu o różnych poziomach trudności. Program miał również wykorzystywać połączenie z relacyjną bazą danych, znajdującą się w kontenerze, do zapisywania zmierzonych czasów przechodzenia przez użytkowników labiryntów oraz budowania na ich podstawie tabeli rankingowych dla poszczególnych poziomów.

1.3 Podsumowanie projektu

Na zrealizowanie projektu przeznaczone zostały około 3 miesiące. Prace zaczęły się od przygotowania dwóch dokumentów – specyfikacji funkcjonalnej oraz specyfikacji implementacyjnej.

Pierwszy z dokumentów opisuje funkcjonalność (opis działania i interakcje z użytkownikiem) oraz wygląd ekranów programu. Zawiera również scenariusz uruchomienia oraz opis przewidywanych sytuacji wyjątkowych i sposobów testowania. Praca nad tą specyfikacją pozwoliła mi uporządkować i sprecyzować jak funkcjonować będzie aplikacja i jakich problemów związanych z jej działaniem mogę się spodziewać.

Specyfikacja implementacyjna z kolei zawiera wiadomości dotyczące tworzenia aplikacji i pracy nad nią. Znajdują się w niej informacje o wykorzystywanych narzędziach i technologiach oraz niezbędnych algorytmach. Zawiera także opis klas i pakietów oraz informacje o bazie danych i kontenerze, w którym ma się ona znajdować. Praca nad tym dokumentem umożliwiła mi lepsze zrozumienie połączenia aplikacji z bazą danych oraz rozplanowanie struktury zarówno programu jak i kontenera z bazą danych.

Po wykonaniu obu specyfikacji przystąpiłam już do pracy nad kodem programu oraz bazą danych. Kolejne części aplikacji (punkty opisane w harmonogramie pracy) dodawałam do repozytorium zazwyczaj co tydzień w piątek, jednak zdarzało się, że w ciągu tygodnia również umieszczałam nowe fragmenty lub wprowadzałam poprawki. W początkowej fazie realizacji projektu miałam drobne problemy ze zdążeniem wykonania zaplanowanych w harmonogramie punktów, jednak ostatecznie udało mi się wykonać wszystkie elementy projektu na czas.

Oprócz wykonywania kolejnych elementów projektu moim zadaniem było również informowanie w formie cotygodniowej wiadomości mailowej opiekuna mojego projektu o postępach prac. Na dzień wysyłania wiadomości wybrałam niedzielę i starałam się dotrzymać tego terminu, jednak niestety zdarzyło mi się wysłać wiadomość dopiero w poniedziałek.

2 Aplikacja

2.1 Narzędzia i technologie

Pisanie kodu aplikacji w języku Java odbywało się przy wykorzystaniu zintegrowanego środowiska programistycznego **IntelliJ IDEA Community Edition 2020.3.3**. Dodatkowym narzędziem

wspierającym pracę nad programem był **Maven**. Oprogramowaniem służącym do realizacji konteneryzacji był **Docker**, a systemem do zarządzania relacyjnymi bazami danych – **PostgreSQL**. W celu realizacji połączenia aplikacji z bazą danych wykorzystany został framework o nazwie **Hibernate**, a sama aplikacja została wykonana w technologii **JavaFX**. Pliki związane z projektem przechowywane były w repozytorium wydziałowym w **Projektorze**, a systemem kontroli wersji został **Git**.

2.2 Moduły i struktura aplikacji

Program został zbudowany ze znacznie większej ilości klas niż było to zaplanowane, jednak ogólna struktura pozostała ta sama. Ostatecznie aplikacja składa się z 5 pakietów – dodany został pakiet *database*, który zawiera klasy odzwierciedlające obiekty bazy danych oraz klasę służącą do łączności z bazą danych. W skład programu wchodzą następujące pakiety i klasy:

- **main**
  - **Main** – klasa główna rozszerzająca klasę Application, z niej następuje uruchomienie głównego okna aplikacji i próba połączenia z bazą danych.
  - **AppRunner** – klasa uruchamiająca aplikację.
- **viewManager**
  - **ViewManager** – klasa odpowiadająca za okno menu głównego, zarządza przyciskami i obszarami pojawiającymi się po ich naciśnięciu.
  - **GameViewManager** – klasa zarządzająca oknem z grą w przechodzenie labiryntu, odpowiada również za interakcję z graczem oraz pomiar czasu. Zawiera prywatną klasę rozszerzającą AnimationTimer, w której odbywa się uruchamianie poruszania się postacią.
- **GUIComponents**
  - **MenuButton** – klasa rozszerzająca Button, służy do tworzenia przycisków menu głównego.
  - **MenuSubscene** – klasa rozszerzająca Subscene, jest to model obszaru pojawiającego się po naciskaniu przycisków.
  - **HelpSubscene** – klasa rozszerzająca MenuSubscene, dodatkowo zawiera elementy dotyczące obszaru z pomocą/opisem gry. Posiada również klasę prywatną rozszerzającą AnimatiomTimer animującą poruszającą się postać.
  - **PlaySubscene** – klasa rozszerzająca MenuSubscene, zawiera wybór poziomu, miejsce na wpisanie pseudonimu oraz przycisk uruchamiający grę. Korzysta również z prywatnej klasy dziedziczącej po AnimationTimer, która realizuje animację na tym obszarze.
  - **LeaderboardSubscene** – klasa rozszerzająca MenuSubscene, reprezentuje obszar z tabelami rankingowymi.
  - **AlertBox** – klasa rozszerzająca Stage, używana jest jako okienko z komunikatem, który należy potwierdzić.
  - **ArrowButton** – klasa rozszerzająca Button, odpowiada za przełączanie tabeli rankingowych.

  - **DIFFICULTY** – klasa typu enumarate, zawiera informacje powiązane z poziomem gry, takie jak wymiary labiryntu czy szybkość postaci.
  - **LevelPicker** – klasa rozszerzająca HBox, służy do stworzenia listy poziomów do wyboru.
  - **MenuTextField** – klasa rozszerzająca TextField, jest to pole gdzie użytkownik wpisuje swój pseudonim.
  - **InfoLabel** – klasa rozszerzająca Label, za jej pomocą umieszczane są napisy.
  - **RankingLabel** – klasa rozszerzająca InfoLabel, używana jest do umieszczania napisów na obszarze z tabelami rankingowymi.
  - **RankingTable** – klasa reprezentująca tabelę rankingową.

- **gameComponents**
  - **MazeGenerator** – klasa służąca do generacji labiryntu o odpowiedniej wielkości.
  - **Dog** – klasa reprezentująca postać, którą gracz steruje podczas gry, zawiera również mechanikę poruszania.
  - **Cell** – klasa odzwierciedlająca komórkę siatki labiryntu.
  - **EndGameSubscene** – klasa dziedzicząca po MenuSubscene, jest to obszar pojawiający się po dotarciu postacią do drzwi końcowych. Wyświetla wynik gracza oraz umożliwia przejście do menu głównego i ponowne rozpoczęcie gry na tym samym poziomie. Zawiera prywatną klasę dziedziczącą po AnimationTimer, która odpowiada za animację postaci na tym obszarze.
  - **GameLabel** – klasa reprezentująca górny pasek w oknie gry z informacjami dotyczącymi aktualnej gry (pseudonim gracza, aktualny czas gry, najlepszy czas dla aktualnego poziomu oraz poziom gry).

- **database**
  - **DatabaseManager** – klasa zarządzająca połączeniem z bazą danych, obsługuje pobieranie informacji oraz ich zapis.
  - **Result** – klasa abstrakcyjna, umożliwia działania na wynikach z każdego poziomu gry jako na wyniku ogólnym. Klasy po niej dziedziczące wykorzystywane są do mapowania obiektów na bazę danych – każda dotyczy innego poziomu gry:
    * **EasyResult**
    * **MediumResult**
    * **HardResult**

Podczas pracy nad programem wykorzystywany był **Maven**, dlatego dodatkowo stworzony został plik **pom.xml**, który zawiera konfigurację projektu.

Aplikacja korzysta również z wielu zasobów związanych z grafiką, które znajdują się w folderze *resources* – są to obrazy PNG, czcionka w pliku z rozszerzeniem TTF oraz opracowanie graficzne paska przewijania na obszarze z tablicami rankingowymi zawarte w pliku scroll.css (napisane w języku CSS).

W celu realizacji kontenera z bazą danych wykorzystywane są następujące pliki:

- **Dockerfile** – plik zawierający informacje niezbędne do stworzenia obrazu, z którego tworzony będzie wykorzystywany kontener z bazą danych.

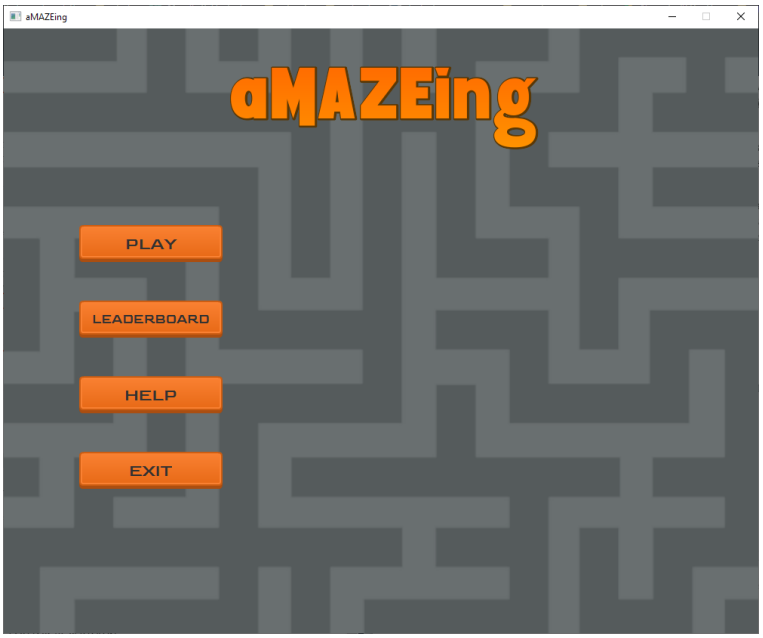
- **docker-compose.yml** – plik wykorzystywany do tworzenia kontenera z obrazu.
- **1-createUser.sql** – skrypt SQL, w którym tworzona jest baza danych *labirynt_database* oraz użytkownik *labirynt_user*, z których korzysta aplikacja.
- **2-createTables.sql** – skrypt SQL, który odpowiada za stworzenie tabel dla wszystkich poziomów trudności oraz funkcji, które ograniczają wielkość tabel do 20 wierszy. Każda z trzech tabel (*EASY, MEDIUM, HARD*) zbudowana jest z 3 kolumn – „id", „nickname" i „time". Numer id jest przypisywany automatycznie, pseudonim gracza i jego wynik czasowy przekazywany jest natomiast przez aplikację.

Aby zrealizować połączenie aplikacji z bazą danych wykorzystywana jest technologia **Hibernate**, dlatego dodany został plik **hibernate.cfg.xml**, który zawiera konfigurację połączenia z bazą danych.

2.3 Ekrany aplikacji

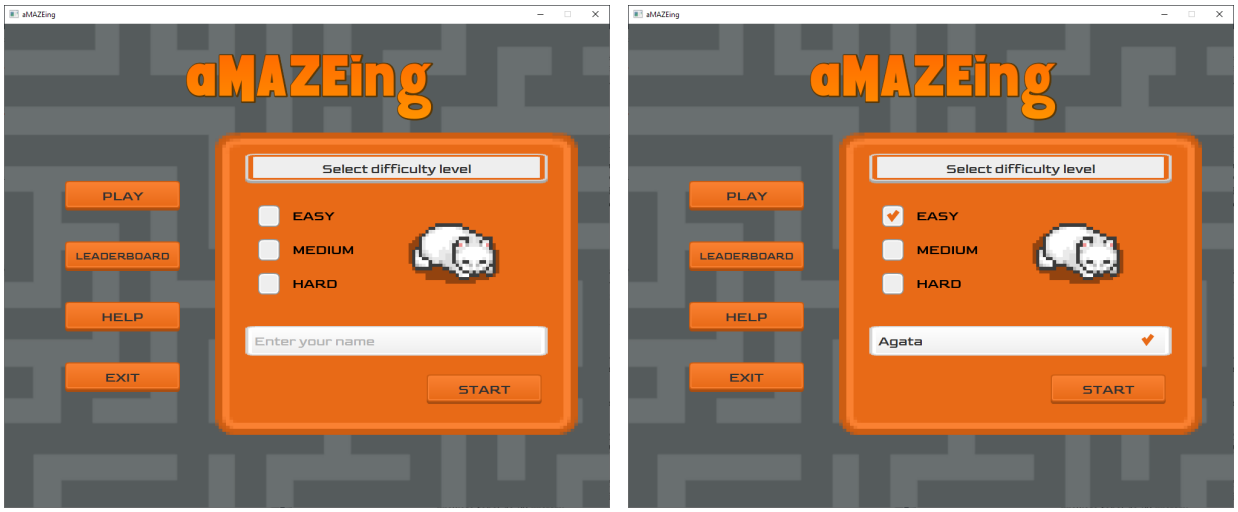
Aplikacja posiada dwa ekrany:

- **Menu główne** – główny ekran aplikacji, z jego poziomu można przeczytać opis gry, zobaczyć tabele rankingowe oraz rozpocząć grę.



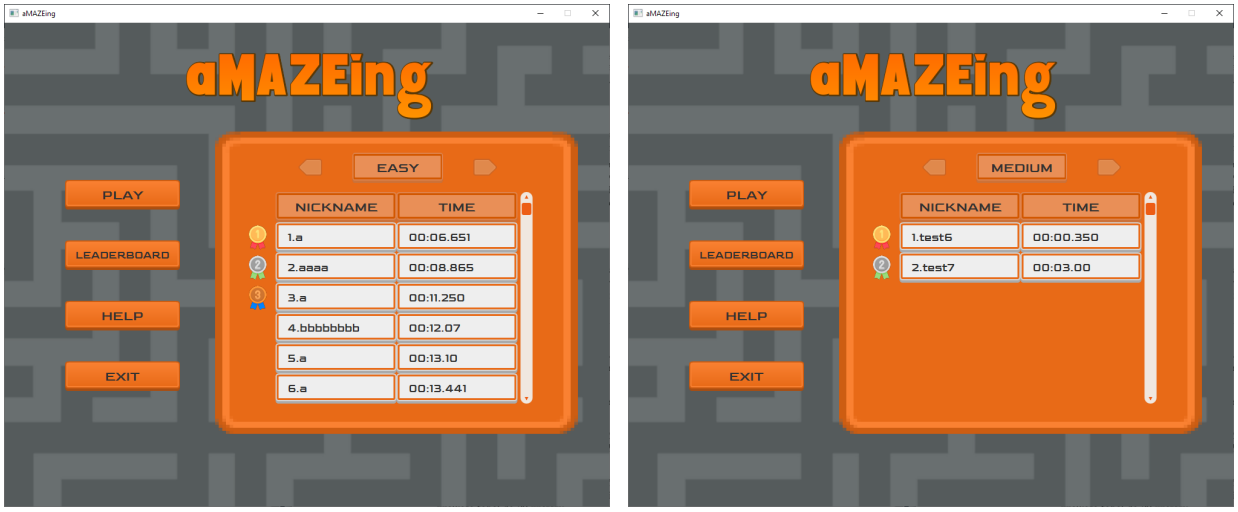
Rysunek 1: Ekran menu głównego

Przycisk EXIT wyłącza aplikację, natomiast po naciśnięciu przycisków PLAY, LEADERBOARD i HELP po prawej stronie pojawia się obszar z odpowiednią zawartością:



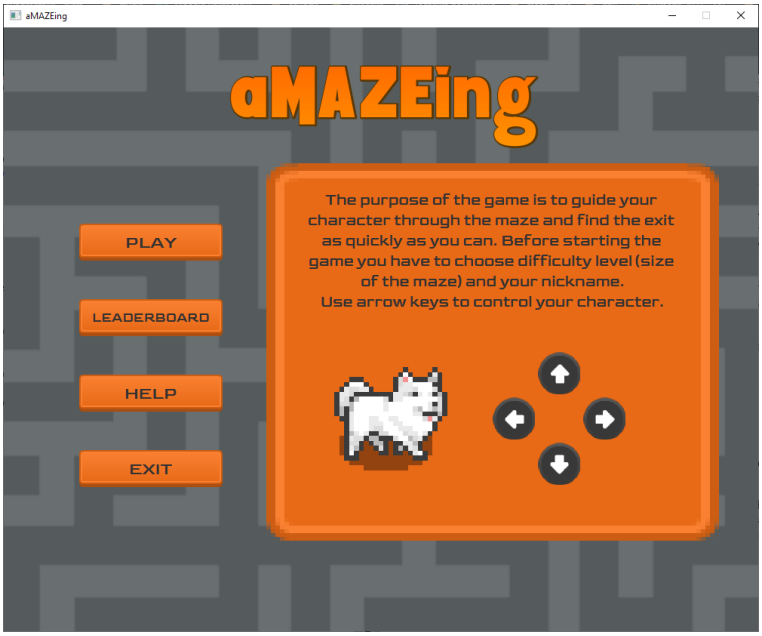
(a) Bez uzupełnionych danych (b) Z wybranym poziomem i wpisanym pseudonimem

Rysunek 2: Ekran menu głównego z obszarem po naciśnięciu przycisku PLAY



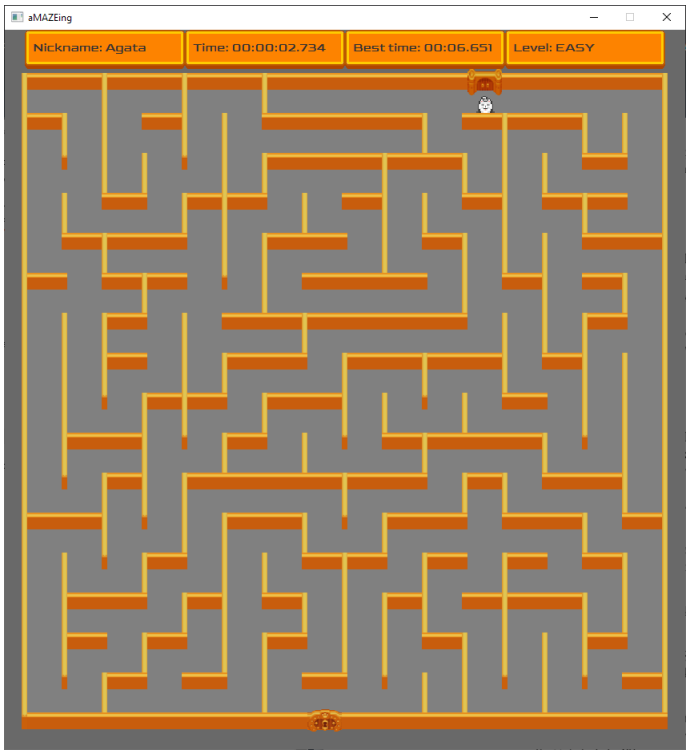
(a) Tabela dla poziomu łatwego (b) Tabela dla poziomu średniego

Rysunek 3: Ekran menu głównego z obszarem po naciśnięciu przycisku LEADERBOARD

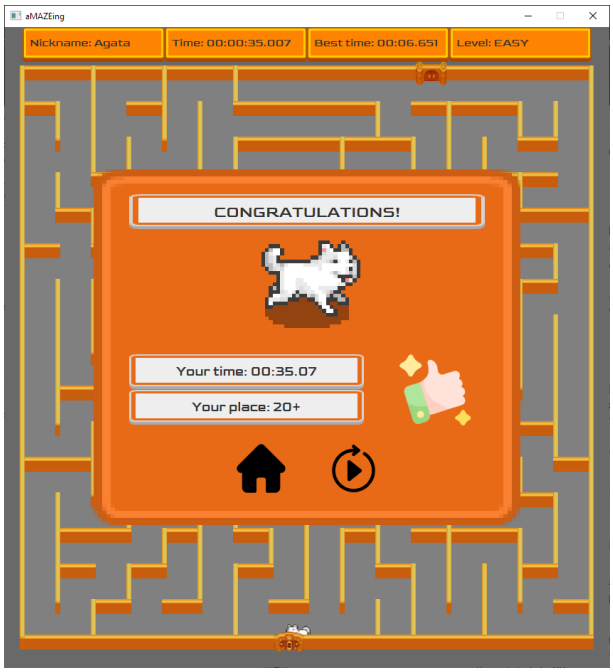


Rysunek 4: Ekran menu głównego z obszarem po naciśnięciu przycisku HELP

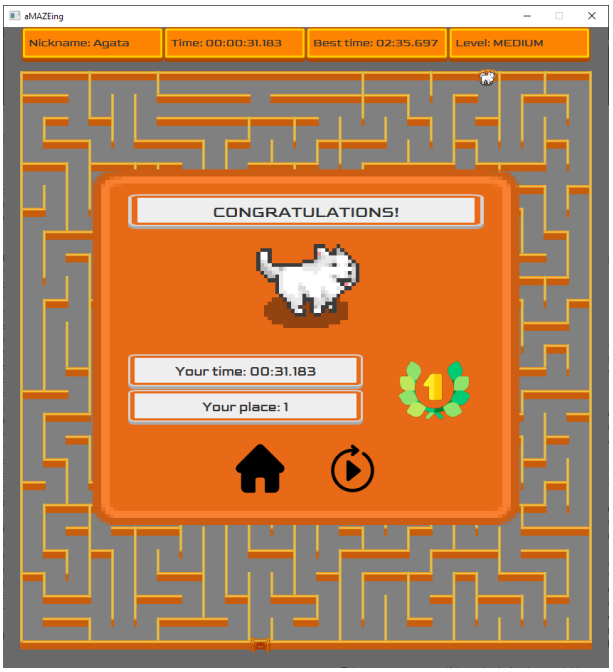
- **Plansza z grą** – pojawia się po wybraniu poziomu, wpisaniu pseudonimu (należy zatwierdzić Enterem) i naciśnięciu przycisku START. Po dojściu do końca labiryntu pojawia się obszar z wynikiem oraz przyciskami umożliwiającymi powrót do menu głównego i ponowne rozpoczęcie gry na tym samym poziomie trudności.



Rysunek 5: Ekran planszy z grą (poziom łatwy labiryntu)



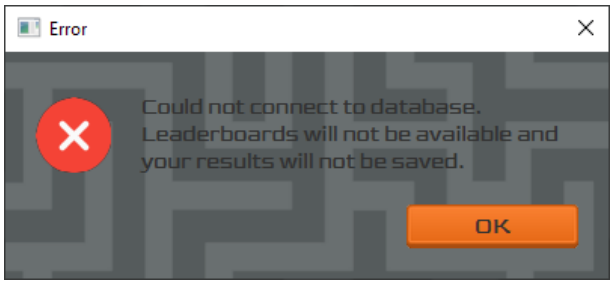
(a) Gdy zdobyte zostanie inne miejsce niż 1



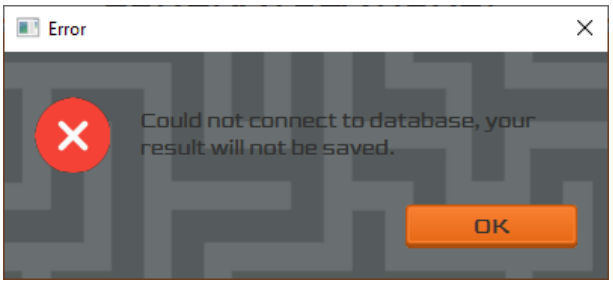
(b) Gdy zdobyte zostanie miejsce 1

Rysunek 6: Ekran planszy z grą z obszarem końca gry

Dodatkowo, gdy następuje problem z połączeniem z bazą danych wyświetlone zostaje okienko z odpowiednim komunikatem:



(a) Komunikat o niemożności połączenia z bazą danych



(b) Komunikat o niemożności zapisania wyniku w bazie danych

Rysunek 7: Ekrany komunikatów

2.4 Uruchomienie aplikacji

Aby uruchomić program należy, będąc w folderze *target*, w konsoli wpisać:

```
java -jar .\aMAZEing.jar.
```

Aby aplikacja mogła korzystać z bazy danych (pobierać i zapisywać wyniki) należy włączyć kontener z bazą danych przed uruchomieniem aplikacji – uruchomienie w trakcie działania nie będzie

brane pod uwagę przez aplikację. Bez połączenia z bazą danych program zachowuje swoją funkcjonalność z wyjątkiem prezentowania tabel rankingowych oraz zapisywania wyników (o czym będą informować komunikaty).

Po wpisaniu powyższej komendy nastąpi próba połączenia z bazą danych, a następnie pojawi się okno z menu głównym (*Rysunek 1*). W przypadku gdy nie uda się połączyć z bazą danych wyświetlone zostanie dodatkowo okienko z odpowiednim komunikatem (*Rysunek 7a*), który należy potwierdzić przyciskiem OK, którego naciśnięcie spowoduje jego zamknięcie. Z poziomu menu głównego można naciskać przyciski **PLAY**, **LEADERBOARD**, **HELP** oraz **EXIT**. Naciśnięcie przycisku EXIT spowoduje wyłączenie aplikacji. Pozostałe przyciski spowodują wyświetlanie lub chowanie odpowiednich obszarów po prawej stronie okna. Obszar po naciśnięciu HELP zawiera opis gry i sposób sterowania postacią poruszającą się po labiryncie (*Rysunek 4*). Po wybraniu przycisku LEADERBOARD nowy obszar zawiera tablicę rankingową z wynikami czasowymi i pseudonimami graczy (*Rysunki 3a i 3b*). Aby obejrzeć tablice dla innych poziomów należy przełączać je za pomocą przycisków po lewej i prawej stronie od etykiety z nazwą poziomu. Tablicę można przewijać za pomocą paska przewijania znajdującego się po jej prawej stronie. Gdy nie udało się połączyć z bazą danych podczas uruchamiania wyświetlone zostają jedynie nagłówki tablic. Ostatni obszar – pojawiający się po naciśnięciu PLAY – zawiera listę z poziomami trudności do wyboru, pole, w które należy wpisać pseudonim (maksymalnie 20 znaków, mogą być znaki polskie oraz inne symbole) oraz przycisk **START**, którego naciśnięcie powoduje przejście do ekranu z grą (*Rysunek 5*). Aby móc rozpocząć grę należy wybrać poziom trudności (klikając na biały kwadrat obok nazwy poziomu) oraz wpisać pseudonim i zatwierdzić przyciskiem Enter. Po wykonaniu tych kroków naciśnięcie przycisku START spowoduje rozpoczęcie gry i przejście do okna z labiryntem. W przypadku niewybrania poziomu lub niewpisania pseudonimu (lub niezatwierdzenia Enterem) po naciśnięciu przycisku START nic się nie stanie.

Gdy pojawi się ekran z labiryntem rozpocznie się pomiar czasu przechodzenia gry. Aktualny czas jest wyświetlany na pasku znajdującym się nad labiryntem, pasek zawiera również pseudonim gracza, najlepszy czas dla danego poziomu oraz nazwę poziomu. Najlepszy czas jest pokazany w przypadku, gdy aplikacja uzyskała połączenie z bazą danych w trakcie uruchamiania, w przeciwnym razie wyświetlone zostaje *00:00.000*. Zadaniem gracza jest doprowadzenie postaci (pieska) od drzwi startowych do drzwi końcowych w jak najkrótszym czasie. Gdy mu się to uda wyświetlony zostaje obszar z wynikiem (czasem oraz zajmowanym miejscem) i przyciskami powrotu do menu głównego (domek) oraz ponownej gry na tym samym poziomie (okrągły przycisk) (*Rysunki 6a i 6b*). W przypadku niepołączenia programu z bazą danych miejsce będzie wynosiło „0”, a gdy połączenie jest a uzyskane miejsce jest większe niż 20 (tylko tyle miejsc jest wyświetlanych w tabeli rankingowej), to wyświetlane jest „20+”. Gdy zapis wyniku w bazie danych nie powiedzie się, to wyświetlony zostaje komunikat o tym informujący (*Rysunek 7b*). W przypadku naciśnięcia przycisku gry ponownej obszar końca gry znika, i pojawia się nowy labirynt do przejścia. Natomiast naciśnięcie przycisku powrotu do menu głównego spowoduje zamknięcie okna planszy z grą, a pojawienie się okna z menu głównym. Jeśli podczas gry okno zostanie zamknięte (za pomocą przycisku „zamknij” X u góry okna), to nastąpi przejście do menu głównego, a wynik nie zostanie zapisany.

3 Informacje o napotkanych problemach

Największym problemem podczas pracy nad projektem okazało się wykorzystanie konteneryzacji, gdyż na początku miałam trudności ze zrozumieniem jak ma wyglądać współdziałanie aplikacji z bazą danych w kontenerze. Aby zrozumieć działanie tego połączenia w ramach ćwiczeń wykonałam krótki program realizujący zapisanie i pobranie danych z bazy danych, przy czym wykorzystałam

zwykłe połączenie przez JDBC. Wykonanie testowego obrazu Dockera i kontenera również pomogło mi lepiej zorientować się w tej dziedzinie. Dzięki wiadomości Pana Magistra Pawła Zawadzkiego, w której otrzymałam opis i wyjaśnienie jak takie połączenie można zrealizować, udało mi się zrozumieć jak wykonać tę część projektu w sposób poprawny i ostatecznie zdecydowałam się skorzystać z mechanizmu mapowania Hibernate. Framework ten pozwala na dość wygodne operowanie na obiektach bazy danych. Pan Magister zwrócił mi także uwagę na zadbanie o bezpieczeństwo bazy danych oraz zabezpieczenie programu w przypadku problemów z połączeniem z bazą danych, co oczywiście wzięłam pod uwagę podczas dalszej pracy nad projektem.

Podczas pracy nad implementacją algorytmu generowania labiryntów (algorytm randomizowanego przeszukiwania w głąb) natknęłam się na problem powstawania długich bardzo mało rozgałęzionych korytarzy. Sytuacja ta wydawała mi się niewłaściwa, gdyż gdy rozgałęzień jest mało i są bardzo krótkie, to rozwiązywanie łamigłówki polegającej na wybieraniu właściwej drogi traci sens. Z tego powodu zdecydowałam się na drobną modyfikację algorytmu – gdy do korytarza zostanie dobudowanych 10 komórek pod rząd, to w następnym kroku rozpocznie się budowa nowego korytarza od połowy tego dziesięciokomórkowego odcinka. Modyfikacja ta poprawiła poziom rozgałęzienia labiryntu w znaczny sposób.

Miałam również drobne problemy z grafiką dotyczącą labiryntu i postaci oraz z mechaniką poruszania się postaci. Problematyczne było dla mnie by postać wyświetlała się przed lub za odpowiednimi ścianami komórek. Ponieważ wyświetlanie to zależy od kolejności dodawania obrazków do panelu, to udało mi się odpowiednio uzależnić numer indeksu obrazka postaci od miejsca, w którym się znajduje, dzięki czemu uzyskałam właściwe wyświetlanie. Problem z mechaniką poruszania dotyczył głównie uniemożliwienia postaci przechodzenia przez ściany lub wchodzenia na części ścian. Problem ten miał związek z poprzednim, gdyż trzeba było wziąć pod uwagę czy postać ma się znajdować przed czy za ścianą i jak daleko może „wejść" na ścianę lub się za nią „schować".

Innym problemem graficznym była tabela rankingowa, gdyż wbudowana w JavęFX klasa TableView była dla mnie bardzo niewygodna do modyfikacji i dopasowania do reszty aplikacji. Dlatego ostatecznie tablicę wykonałam ręcznie za pomocą klas HBox, VBox oraz RankingLabel, a pasek przewijania za pomocą klasy ScrollBar, której styl zmodyfikowałam zewnętrznym plikiem w języku CSS.

4 Podsumowanie

4.1 Podsumowanie produktu końcowego

Wykonana przeze mnie aplikacja zawiera wszystkie elementy wyszczególnione w zakresie pracy znajdującym się na ISODzie. Dodałam dodatkową nieplanowaną funkcjonalność – na obszarze końca gry znajduje się możliwość wyboru między przejściem do menu głównego a ponowną grą na tym samym poziomie. Wygląd ekranów programu znacznie różni się od tych zaprezentowanych przeze mnie w specyfikacji funkcjonalnej, jednak w dokumencie zawarłam jedynie prototypy, a podczas pracy nad aplikacją postanowiłam je dopracować. Znaczna różnica występuje również w strukturze programu, stworzyłam o wiele więcej klas niż zaplanowałam. Podczas opracowywania przewidywanej budowy aplikacji spodziewałam się, że prawdopodobnie ostatecznie stworzę więcej klas, gdyż podczas planowania programu trudno jest dokładnie przewidzieć jego dokładną strukturę. Nie spodziewałam się jednak, że nowych klas będzie aż tak dużo. Większość nieplanowanych przeze mnie klas dotyczy komponentów graficznych, co ma związek z dopracowywaniem przeze mnie wyglądu ekranów aplikacji w trakcie pracy nad kodem.

4.2 Podsumowanie pracy nad realizacją projektu

Dzięki realizacji tego projektu poznałam konteneryzację oraz poszerzyłam wiedzę o bazach danych i tworzeniu aplikacji. Docker wydaje mi się bardzo przydatnym i wygodnym narzędziem zarówno do pracy nad programami, jak i do uruchamiania w kontenerach aplikacji lub ich części. Nigdy wcześniej nie miałam również okazji do połączenia bazy danych z aplikacją, a wydaje mi się to przydatną umiejętnością. Dzięki harmonogramowi przygotowanemu przez Pana Magistra Pawła Zawadzkiego wiedziałam jak rozplanować pracę nad projektem, a sam harmonogram mi odpowiadał.