UNIWERSYTET GDAŃSKI

WYDZIAŁ MATEMATYKI, FIZYKI I INFORMATYKI

Sebastian Czyżewski

Kierunek: Informatyka ogólnoakademicka
Specjalność: Aplikacje internetowe i bazy danych
Numer Albumu: 260964

Aplikacja webowa wykorzystująca algorytmy do optymalizacji czynienia postępów w grze Animal Crossing: New Horizons

Praca licencjacka napisana
pod kierunkiem dr Piotra Arłukowicza

Gdańsk 2021

Spis treści

1. Streszczenie **2**

2. Opis problemu **3**

3. Analiza aplikacji **4**
3.1. Funkcjonalność 4
3.2. Użytkownicy 4
3.3. Technologie 5
3.4. Koncepcja graficzna 5

4. Analiza badań własnych **8**

5. Implementacja **9**
5.1. Przechowywanie danych 9
5.2. Główne klasy 10
5.3. Użytkownicy, logowanie i rejestracja 10
5.4. Algorytm 12
5.5. Front-end 13

6. Zakończenie **14**

7. Bibliografia **15**

8. Spis rysunków **16**

1. Streszczenie

Zakres pracy obejmuje projekt oraz implementację aplikacji webowej, stworzonej dla graczy Animal Crossing: New Horizons. Animal Crossing jest grą wydaną przez Nintendo na konsole Nintendo Switch. Podstawową funkcją aplikacji jest zbiór informacji na temat występujących w grze zwierząt, które gracze mogą kolekcjonować lub sprzedawać, aby zarobić dostępną w grze walutę. Informacje te ułatwiają łapanie konkretnych gatunków. Użytkownicy mogą też modyfikować listy zwierząt złapanych i tych jeszcze niezebranych, aby na bieżąco śledzić postępy swojej kolekcji. Ponadto, użytkownicy mają możliwość skorzystania z algorytmu, który oblicza, jakie czynności są aktualnie najefektywniejsze dla skompletowania kolekcji lub zarobienia pieniędzy. Algorytm opiera się na tym, że każdy gatunek zwierzęcia ma do siebie przypisaną rzadkość. Nigdzie nie ma natomiast informacji na temat jak dokładnie te rzadkości działają, dlatego też zostały przeprowadzone badania mające na celu ustalenie dokładnych zależności między rzadkościami gatunków. Program bierze pod uwagę bezpieczeństwo użytkowników, dlatego ma zaimplementowany system szyfrowania haseł oraz potwierdzania adresów e-mail. Koncepcja graficzna aplikacji opiera się na prostocie i schludności, ze względu na założenie twierdzące, że potencjalny użytkownik chce spędzić jak najmniej czasu na poszukiwaniu pożądanych informacji. Technologie zastosowane w programie zostały wybrane z uwagi na swoją popularność jak i łatwość w obsłudze. Projekt oparty jest na technologii Java, Spring, PostgreSQL, HTML5 oraz CSS.

2. Opis problemu

Animal Crossing: New Horizons jest to wydana przez Nintendo gra dostępna na platformie Nintendo Switch, która zabiera graczy na początkowo bezludną wyspę na środku morza. Głównym celem graczy jest rozbudowa infrastruktury i zaludnienie wyspy, tak aby ostatecznie przekształcić ją w niewielkie miasteczko. Aby zrealizować ten cel potrzebna jest między innymi dostępna w grze waluta. Animal Crossing udostępnia graczom wiele czynności, dzięki którym można zarobić pieniądze, a jedną z nich jest łapanie, a następnie sprzedawanie zwierząt występujących na wyspie (tzw. *Critters*). Zwierzęta te dzielą się na 3 kategorie: ryby, robaki i stworzenia morskie. Dodatkowo po złapaniu przedstawiciela konkretnego gatunku, można go jednorazowo umieścić w dostępnym na wyspie muzeum. Dzięki temu, celem wielu graczy stało się zebranie wszystkich dostępnych w grze gatunków. Animal Crossing jednak nie ułatwia tego zadania graczom. Gra pobiera z konsoli aktualną datę oraz godzinę, a występowanie wszystkich gatunków jest ograniczone czasowo. Dla przykładu, ryby z gatunku *Nibble fish* występują tylko od kwietnia do września w godzinach od 9:00 do 16:00. Dodatkowo ryby występują tylko w konkretnych zbiornikach wodnych, przez co ryby dostępnej w rzece nie można złowić w morzu. Gracz postawiony w takiej sytuacji może łatwo przekroczyć termin dostępności danego gatunku, przez co może być zmuszony do odczekania kilku miesięcy na następne pojawienie się tego gatunku na wyspie. Głównym celem zaprojektowanej przeze mnie aplikacji jest zapobieganie takim sytuacjom. Program udostępnia listę wszystkich dostępnych zwierząt, wraz z wszystkimi potrzebnymi do ich złapania informacjami. Użytkownik może oznaczyć konkretne gatunki jako dodane do muzeum, aby z łatwością śledzić postępy swojej kolekcji. Dodatkowo aplikacja udostępnia dwa algorytmy, które podpowiadają użytkownikowi, na czym ma się skupić, aby uzupełnić swoją kolekcję lub szybciej zarobić pieniądze.

3. Analiza aplikacji

3.1. Funkcjonalność

Aplikacja dzieli się na trzy główne funkcje. Pierwszą z nich jest zaprezentowanie listy dostępnych w grze zwierząt wraz z ich informacjami. Wszystkie trzy typy zwierząt (ryby, robaki i stworzenia morskie) mają cechy takie jak nazwa, cena, rzadkość oraz lista miesięcy i godzin, w których dany gatunek jest dostępny. Każdy typ ma też kilka dodatkowych cech. Niektóre robaki nie pojawiają się, kiedy w grze pada deszcz oraz pojawiają się tylko, kiedy spełnione są specjalne warunki. Większość lata w powietrzu lub siedzi na kwiatach, ale aby złapać konkretne gatunki trzeba np. potrząsnąć drzewem, aby spadł z niego ul i wyleciały z niego pszczoły, albo zostawić na ziemi zgniłe jedzenie, aby zaczęły się dookoła niego zlatywać muchy. Morskie stworzenia i ryby przed złowieniem pojawiają się w grze jako cień w wodzie, który w zależności od gatunku różni się rozmiarem. Ryby dzielą się na te, które można złowić w rzece, morzu oraz stawie. Dodatkowo konkretne gatunki morskie są dostępne tylko na molo, a rzeczne tylko w ujściu lub źródle rzeki. Wszystkie te informacje wraz z przedstawiającymi zwierzę ikonami są zaprezentowane w odpowiednich tabelach. Program daje też możliwość dodawania, edycji oraz usuwania gatunków w razie aktualizacji gry lub po popełnieniu błędu przy dodawaniu. Użytkownicy mogą swobodnie przerzucać gatunki z listy niezebranych do listy zgromadzonych i z powrotem, aby na bieżąco śledzić postępy swoich kolekcji. W aplikacji występują też dwa algorytmy. Jeden z nich, po podaniu warunków panujących na wyspie (czy pada deszcz oraz czy tego dnia na wyspie jest handlarz, któremu można sprzedać ryby za półtora raza więcej pieniędzy) oblicza, jaka czynność (np. łowienie ryb w stawie) poskutkuje zarobieniem największej ilości pieniędzy. Drugi algorytm, po podaniu tych samych informacji obliczy, na której czynności powinien skupić się gracz, aby uzupełnić swoją kolekcję. Ze względu na to, że Animal Crossing nie obsługuje polskiej wersji językowej, wszystko na stronie jest napisane w języku angielskim.

3.2. Użytkownicy

Program daje możliwość logowania oraz rejestracji nowych użytkowników. Bezpieczeństwo jest jednym z najważniejszych niefunkcyjnych aspektów wszelkich aplikacji [1]. Nawet najprostsza strona internetowa powinna dbać o bezpieczeństwo i zachowanie prywatności swoich użytkowników. Z tego powodu w aplikacji istnieje system szyfrujący hasła, aby w bazie danych nie przechowywać ich jako

tekst jawny. Po rejestracji, użytkownik ma piętnaście minut na potwierdzenie adresu e-mail. Po upłynięciu tego czasu link potwierdzający e-mail wygasa, ale istnieje możliwość wygenerowania i wysłania kolejnego linku. Użytkownicy podzieleni są według ról na dwie grupy: zwykłych użytkowników i administratorów. Zwykli użytkownicy mogą wyświetlać tabele zwierząt, zarządzać swoją kolekcją oraz korzystać z algorytmów. Administratorzy mogą wszystko to, co zwykli użytkownicy, ale dodatkowo mają możliwość dodawania, edytowania i usuwania konkretnych gatunków.

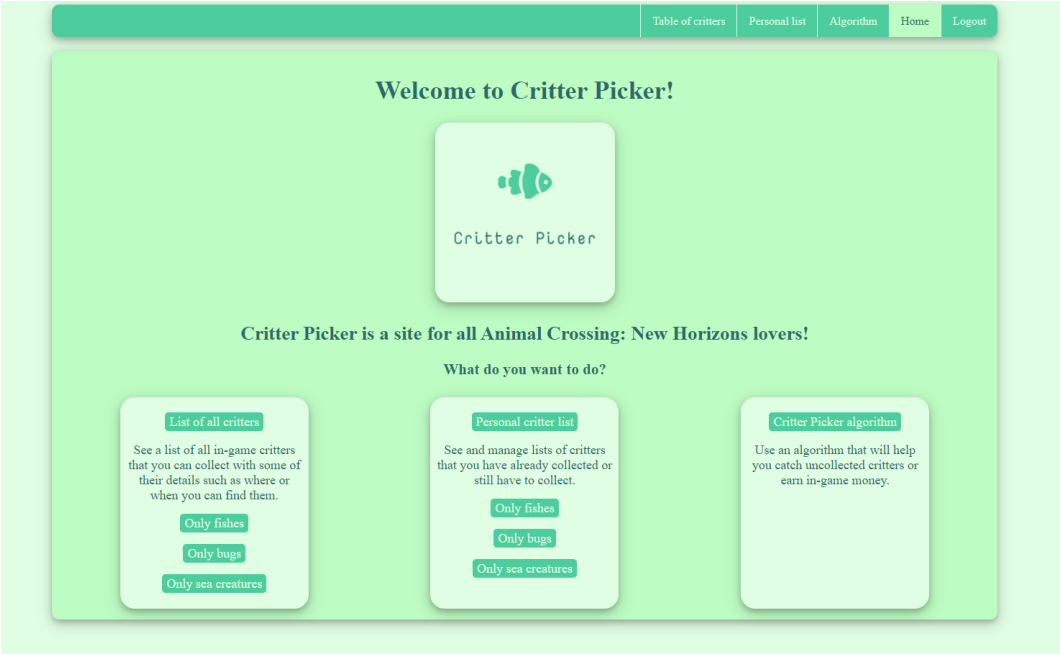
3.3. Technologie

Aplikacja napisana jest głównie w języku programowania Java. Dużą zaletą języka Java jest to, że bardzo ułatwia pracę dzięki wbudowanym funkcjom oraz dużemu zbiorowi bibliotek [2]. Dobrym tego przykładem jest wykorzystana w programie biblioteka Project Lombok, która zasadniczo zwiększa produktywność przy pisaniu klas. W aplikacji wykorzystany jest framework Spring, ponieważ jest on bardzo popularny oraz znacznie upraszcza tworzenie aplikacji webowych, a framework Spring Boot zajmuje się wstępną konfiguracją oraz dostarcza serwer sieciowy Tomcat [3]. Dodatkowo framework Spring Security zapewnia wszystkie narzędzia potrzebne do zadbania o bezpieczeństwo programu oraz zaimplementowania logowania i rejestracji. Za przechowywanie wszelkich danych odpowiada relacyjna baza danych PostgreSQL. Potrafi ona szybko wykonywać zaawansowane funkcje, dzięki czemu jest ona zdecydowanie lepsza od wielu innych aktualnie dostępnych baz danych [4]. Za front-end (tj. pobieranie od i zwracanie do użytkownika danych oraz generowanie widoków) oprócz standardowych języków HTML5 i CSS odpowiedzialny jest także Thymeleaf. Jest on dosyć popularnym silnikiem szablonów, który cechuje się swoją podobną do języka HTML składnią, dzięki czemu jest stosunkowo prosty w obsłudze [5].

3.4. Koncepcja graficzna

Widoki w aplikacji cechują się prostotą. Użytkownicy chcą poświęcić jak najmniej czasu, aby uzyskać pożądane informacje [6]. Kolorystyka strony ogranicza się do kilku odcieni zieleni. Kolor zielony kojarzy się z naturą, co jest adekwatne do tematyki programu. Jak widać na rysunku 1. widok strony dzieli się na znajdujący się na górze pasek nawigacyjny z linkami do wszystkich ważnych podstron oraz zbiornik główny, w którym znajduje się zawartość przeznaczona dla aktualnej strony. Pasek nawigacyjny, zbiornik główny oraz niektóre elementy

mają dodany cień, co sprawia, że całość nabiera trójwymiarowego wyglądu i jest bardziej przejrzysta. Logo widoczne na rysunku 2. zawiera nazwę strony oraz ikonę przedstawiającą rybę. Zostało ono wygenerowane za pomocą strony „hatchful.shopify.com”. Nazwa Critter Picker krótko opisuje przeznaczenie aplikacji — pomaganie użytkownikowi w łapaniu zwierząt w grze.



Rysunek 1. Źródło: Wykonanie własne. Zrzut ekranu ze strony domowej aplikacji.

Rysunek 2. Źródło: Wygenerowane za pomocą strony „hatchful.shopify.com”. Logo strony. Prawa autorskie zezwalają na nieodpłatne i legalne kopiowanie oraz udostępnianie.

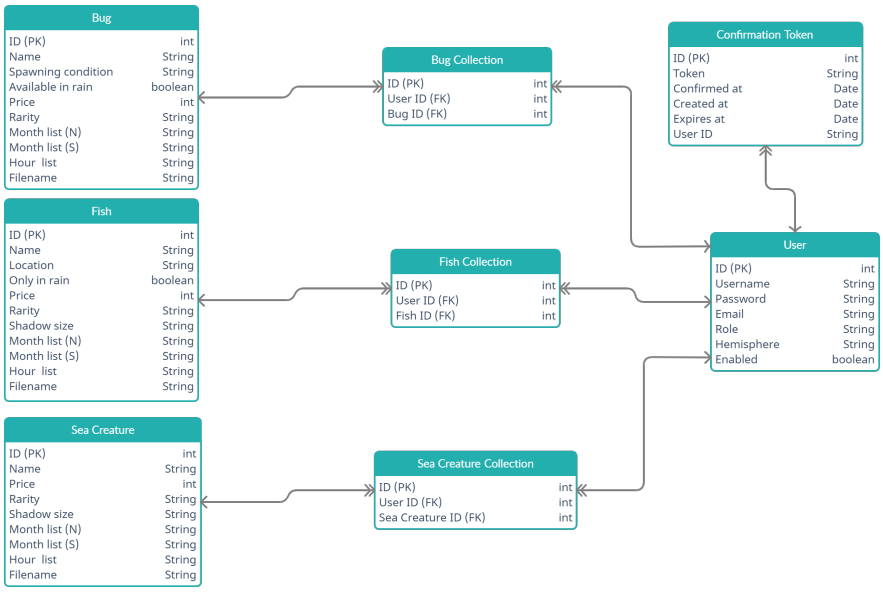
4. Analiza badań własnych

Każde zwierzę dostępne w grze ma przypisaną rzadkość: *common*, *fairly common*, *uncommon*, *scarce* lub *rare*. Z moich obserwacji wynika, że rzadkości gatunków w grze działają w następujący sposób. Do każdej rzadkości przypisana jest pewna liczba. Kiedy na wyspie pojawia się możliwe do złapania zwierzę, jest ono losowane z pewnej puli. Pula ta jest początkowo pusta, a następnie dodają się do niej osobniki każdego dostępnego w danej chwili gatunku w ilości równej liczbie przypisanej do jego rzadkości. Przykładowo, jeżeli w danym zbiorniku o danym czasie byłyby dostępne dwa gatunki, jeden rzadkości *common*, a drugi rzadkości *scarce*, to w puli byłoby kilkanaście lub kilkadziesiąt osobników pierwszego gatunku i tylko kilka osobników drugiego, rzadszego gatunku. W takiej sytuacji, pomimo że szansa na wylosowanie każdego osobnika jest równa, to bardziej prawdopodobne jest wylosowanie ryby z pierwszego gatunku, ponieważ ma on więcej osobników w puli. Dzięki temu szansa na wylosowanie danego gatunku zawsze jest zależna od reszty aktualnie dostępnych zwierząt. W grze ani w internecie nie ma natomiast żadnych informacji odnośnie tych przypisanych do rzadkości liczb, dlatego niezbędne dla prawidłowego funkcjonowania algorytmu było przeprowadzenie odpowiednich badań. W tym celu, złowiono 200 ryb w morzu, w maju, w godzinach 21:00 – 24:00. W danych warunkach pula dostępnych ryb składała się z 13 ryb, w tym 3 rzadkości *common*, 1 *fairly common*, 6 *uncommon*, 2 *scarce* oraz 1 *rare*. Wyniki tego badania były następujące: 91 ryb rzadkości *common*, 18 *fairly common*, 73 *uncommon*, 16 *scarce* i 2 *rare*. Po drobnych zmianach (91 *common* zmieniono na 90, 73 *uncommon* na 72), mających na celu zaokrąglenie ostatecznych wyników, liczby złowionych ryb zostały podzielone przez ilość gatunków danych rzadkości, a następnie podzielone przez 2 czyli największy wspólny dzielnik. W ten sposób otrzymano liczby przypisane do rzadkości. Są one następujące: 15 dla rzadkości *common*, 9 *fairly common*, 6 *uncommon*, 4 *scarce* i 1 *rare*. Liczby te znajdują się w postaci zmiennych na początku klasy `AlgorithmManager`, co umożliwia ich zmianę w razie ponownego przeprowadzenia badań na większą skalę i otrzymania dokładniejszych wyników.

5. Implementacja

5.1. Przechowywanie danych

Za przechowywanie danych w programie odpowiedzialna jest baza danych PostgreSQL. Baza jest skonfigurowana w pliku `application.properties` znajdującym się w folderze `resources`. Tabele `Bug`, `Fish` i `SeaCreature` połączone są relacją wiele do wielu z tabelą `AppUser`. Schemat bazy danych przedstawia rysunek 3. Aktualnie w bazie znajdują się wszystkie dostępne w grze zwierzęta, administrator z nazwą użytkownika ZaQuty68 i hasłem ZaQuty68 oraz dwóch zwykłych użytkowników user1 i user2 z hasłami kolejno: user1password i user2password. Kopia bazy danych została wykonana za pomocą komendy `pg_dump` i umieszczona pod ścieżką `resources/databaseBackup`. Za zarządzanie plikami zawierającymi ikony konkretnych gatunków odpowiedzialna jest klasa `StorageManager` umieszczona w pakiecie `Storage`. Pliki te przechowywane są pod ścieżką `resources/static/upload-dir`. Dane ikony pobrane są ze strony internetowej „animalcrossing.fandom.com” i są objęte prawami autorskimi na podstawie licencji CC-BY-SA, co zezwala na nieodpłatne i legalne kopiowanie oraz udostępnianie pod warunkiem wskazania autora.



Rysunek 3. Źródło: Wykonanie własne. Diagram relacji tabel w bazie danych.

5.2. Główne klasy

W aplikacji w pakiecie `Critters` znajdują się klasy `Fish`, `Bug` oraz `SeaCreature`, które są modelami wszystkich trzech typów dostępnych w grze zwierząt. Posiadają one pola zawierające wszystkie potrzebne do złapania danego gatunku informacje jak i nazwę pliku z ikoną przedstawiającą zwierzę oraz zbiór użytkowników posiadających dany gatunek w swojej kolekcji. Każdy gatunek posiada dwie listy zawierające numery miesięcy, w których jest dostępny, ponieważ różnią się one w zależności od półkuli, na której mieszka gracz. Przy dodawaniu nowego zwierzęcia potrzebne jest jednak podanie tylko jednej listy, ponieważ miesiące dostępności na półkuli południowej są przesuniętymi o pół roku do przodu miesiącami z półkuli północnej. Dzięki adnotacjom `@Entity` oraz `@Table` modele stają się jednostkami umieszczanymi w odpowiednich tabelach w bazie danych. Odpowiadające klasom interfejsy są rozszerzeniami `JpaRepository` dostarczanego przez framework Spring. Powstałe w ten sposób repozytoria umożliwiają zarządzanie jednostkami w bazie danych. Do każdego modelu istnieją także obsługujące je klasy `FishManager`, `BugManager` oraz `SeaCreatureManager`, które zawierają operujące na odpowiednich repozytoriach funkcje, takie jak dodawanie czy edytowanie gatunków. Oprócz tego istnieją też zbliżone do modelów klasy pomocnicze DTO (ang. *Data Transfer Object*), czyli obiekty do transferu danych. Są one potrzebne, ponieważ administratorzy dodając lub edytując gatunki dostarczają między innymi listy miesięcy oraz godzin, w których dane zwierzę jest dostępne, ale baza danych nie może przechowywać listy w pojedynczej kolumnie. Administratorzy w rzeczywistości nie tworzą bezpośrednio dodawanej do bazy jednostki, tylko obiekty DTO, które następnie menedżer przekonwertowuje na odpowiednie modele zmieniając listy w tekst, po czym dodaje je do bazy danych.

5.3. Użytkownicy, logowanie i rejestracja

Logowanie oraz użytkownicy zostali zaimplementowani za pomocą frameworku Spring Security. Plik `WebSecurityConfiguration` znajdujący się w pakiecie `Config` służy do konfiguracji rzeczy związanych z bezpieczeństwem strony, takich jak autoryzacja. Jedną z jego funkcji jest zezwalanie na dostęp wszystkim użytkownikom do stron odpowiedzialnych za logowanie i rejestrowanie, jednocześnie zabraniając dostępu do reszty stron niezalogowanym użytkownikom. Dodatkowo sprawia on, że wyznaczone dla administratorów strony są dostępne tylko dla nich. Logowanie jest tak naprawdę zarządzane całkowicie przez fra-

mework Spring Security, jednak został dla niego zmieniony widok tak, aby pasował do koncepcji graficznej strony. Klasa `AppUser` implementuje dostarczaną przez framework Spring Security klasę `UserDetails`. Jest to spersonalizowany pod potrzeby programu model użytkownika, który oprócz wszystkich potrzebnych do autoryzacji danych zawiera informację o półkuli, na której mieszka gracz oraz zbiory ryb, robaków i stworzeń morskich, które zebrał użytkownik. Pakiet `Registration` zawiera wiele klas służących do rejestracji. Potencjalny użytkownik uzupełniając formularz rejestracyjny tworzy instancję klasy `RegistrationRequest`, którą po walidacji (tj. po sprawdzeniu czy pola spełniają konkretne wymagania) otrzymuje klasa `RegistrationManager`. Aby zostać administratorem należy dodatkowo podać hasło (obecnie: „MakeMeAnAdmin123") ustalone przez zmienną `adminPassword` na początku pliku `RegistrationManager`. Jeżeli ani nazwa użytkownika, ani e-mail nie istnieją już w bazie danych, klasa `RegistrationManager` tworzy nowego użytkownika, szyfrując jego hasło za pomocą dostarczanego przez framework Spring Security systemu szyfrującego `BCryptPasswordEncoder`, a następnie tworzy dla niego instancję klasy `ConfirmationToken`. Taki obiekt zawiera token, czyli unikalny ciąg znaków, oraz daty stworzenia, przeterminowania oraz potwierdzenia tokena. Potem z pomocą klasy `EmailSender` menedżer rejestracji wysyła wiadomość e-mail na podany adres z linkiem potwierdzającym, który zawiera przypisany użytkownikowi token. Kiedy użytkownik przejdzie pod podany link, jeżeli ten jeszcze nie został przeterminowany, jego konto zostaje aktywowane i dopiero wtedy uzyskuje możliwość logowania i dostęp do reszty stron. Jeżeli nie zdąży on przed wygaśnięciem linka lub dane konto jest już aktywowane, dostanie on stosowny komunikat. Użytkownicy mają też możliwość wygenerowania i wysłania kolejnego linka potwierdzającego. Wysyłanie wiadomości e-mail zostało zaimplementowane za pomocą otwartego oprogramowania (ang. *Open Source*) MailDev pobranego ze strony „github.com/maildev/maildev", dlatego wiadomości te nie są tak naprawdę wysyłane do skrzynek pocztowych użytkowników, lecz na adres „localhost:1080/#/". Jest to rozwiązanie tymczasowe, pozwalające na tworzenie wielu użytkowników bez potrzeby tworzenia nowych prawdziwych adresów e-mail. Aby program poprawnie rejestrował użytkowników i nie zgłaszał błędu przy próbie wysłania wiadomości e-mail należy najpierw na urządzeniu mieć zainstalowany MailDev i dopilnować, aby w trakcie rejestracji działał on w tle.

5.4. Algorytm

Algorytm pobiera informacje o tym jaki wariant, zarabianie pieniędzy czy uzupełnienie kolekcji, jest pożądany przez użytkownika oraz czy danego dnia pada deszcz i czy na wyspie jest CJ czyli handlarz, któremu można sprzedawać ryby za półtora raza więcej pieniędzy. Ostatecznie zwraca czynności, które są najbardziej optymalne dla postawionego celu. Początkowo program pobiera listy ryb i morskich stworzeń. W przypadku wariantu pieniężnego są to wszystkie gatunki. W przeciwnym razie dodatkowo pobierane są gatunki nie zebrane do tej pory przez użytkownika. Robaki są z tego algorytmu wyłączone, ponieważ niektóre pojawiają się w grze tylko po spełnieniu konkretnych warunków, co skutecznie utrudnia przewidywanie, jakie gatunki mogą pojawić się na wyspie u gracza. Algorytm następnie pobiera miesiąc daty dzisiejszej, oraz miesiąc daty przesuniętej o pewną ilość dni do przodu. Przesunięcie jest regulowane przez umieszczoną na początku klasy `AlgorithmManager` zmienną `plusDays`, której domyślna wartość wynosi jeden, tak aby algorytm wyliczał optymalne czynności również dla kolejnego dnia. Można jednak wartość tej zmiennej dostosować, tak aby przetestować odpowiedzi algorytmu dla odsuniętych daleko w czasie miesięcy. W następnej kolejności aplikacja dla każdej pozostałej godziny aktualnego dnia oraz wszystkich godzin następnego dnia przygotowuje wcześniej pobrane listy stworzeń, ograniczając je tylko do tych gatunków, które są w danym miesiącu i godzinie dostępne na wyspie, oraz w przypadku ryb rozdzielając je na lokacje, w których występują. Kolejną czynnością algorytmu jest wyliczanie dla każdej z otrzymanych w ten sposób list średniej ceny, w przypadku wariantu pieniężnego, lub średniego priorytetu, w przypadku wariantu kolekcjonerskiego. Średnie ceny są wyliczane za pomocą funkcji `getAveragePriceFish` i `getAveragePriceSeaCreature`. Funkcje te sumują ceny gatunków wymnożone przez przypisane do ich rzadkości liczby, a następnie dzielą otrzymaną sumę przez rozmiar puli tworzonej w taki sam sposób, w jaki opisane jest tworzenie puli w grze w rozdziale 4. Średni priorytet obliczany jest przez sumowanie priorytetów zwierząt z list jeszcze nie zebranych, a następnie dzielenie przez tą samą pulę. Priorytet pojedynczego gatunku wyliczany jest za pomocą wzoru:

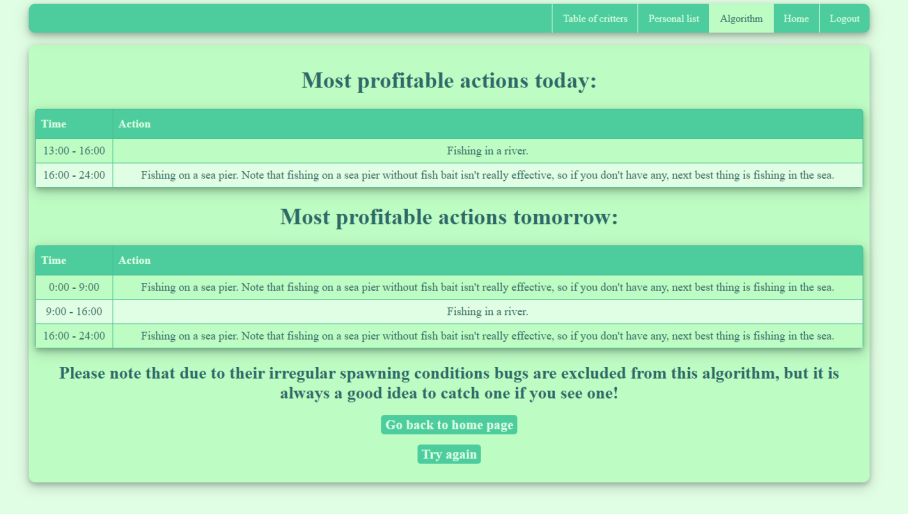
$$priority = x * ((raritySum - x) + (12 - monthsLeft) * 10)$$

Gdzie:

- priority to priorytet gatunku
- x to liczba przypisana do rzadkości danego gatunku

- raritySum to suma liczb przypisanych do wszystkich rzadkości
- monthsLeft to ilość następujących miesięcy, które zostały do zniknięcia gatunku z wyspy (maksymalnie 11)

Po obliczeniu odpowiednich średnich, algorytm wybiera tą największą i tworzy odpowiedź. Kiedy odpowiedzi są stworzone dla wszystkich godzin, program przygotowuje nowe odpowiedzi, łącząc przedziały czasowe dla tych samych czynności. Aplikacja na koniec zwraca przygotowane w ten sposób odpowiedzi użytkownikowi. Przykładowy wynik działania algorytmu przedstawiony jest na rys. 4.



Rysunek 4. Źródło: Wykonanie własne. Zrzut ekranu ze strony z odpowiedzią algorytmu.

5.5. Front-end

Za zarządzanie adresami (ang. endpoint) i zapytaniami (ang. request) odpowiedzialny jest kontroler zdefiniowany w pliku `WebController` znajdującym się w pakiecie `Controller`. Stylistyka stron zawarta jest w jednym pliku o nazwie `styles.css`, w którym zawarte są definicje wyglądu strony zgodne z koncepcją graficzną projektu. Aby style wyświetlały się poprawnie, pierwsza linijka w pliku `application.properties` musi zawierać ścieżkę wskazującą na folder główny projektu. Szablony stron napisane są za pomocą języka HTML5 i wygenerowane za pomocą silnika szablonów Thymeleafa.

6. Zakończenie

W programie jest wiele miejsca na ulepszenia. Zaznaczanie przedziałów czasowych przy dodawaniu i edytowaniu zwierząt jest czasochłonne i niewygodne. W widokach przedstawiających kolekcje użytkownika przydałyby się widoki częściowe, tak aby po przerzuceniu gatunku z jednej listy do drugiej, strona nie odświeżała się całkowicie tylko częściowo. Są to problemy związane silnie z frontendem i prawdopodobnie można by je rozwiązać dodając do projektu technologię JavaScript. Aplikacja nie jest umieszczona na żadnym prawdziwym serwerze, więc działa tylko na localhoście. Dodatkowo przydałby się system wyszukiwania i sortowania zwierząt w odpowiednich tabelach. Zdecydowanym ulepszeniem projektu byłoby też ponowne przeprowadzenie badań, tym razem na większą skalę, aby zwiększyć dokładność algorytmu. Dobrym dodatkiem byłby też system pozwalający użytkownikowi na odzyskanie zapomnianego hasła. Z drugiej strony aplikacja spełnia wszystkie postawione na wstępie cele. Tabele zwierząt zawierają wszystkie ważne informacje o każdym dostępnym w grze gatunku, a algorytm skutecznie ułatwia czynienie postępów w grze. Program działa stabilnie i bez żadnych opóźnień. Ostatecznie projekt zakończył się sukcesem, jednak jest kilka rzeczy, które można by ulepszyć.

7. Bibliografia

[1] L.Spilca, „*Spring Security in Action*", Simon and Schuster, 2020
ISBN: 1617297739, 9781617297731

[2] C.S.Horstmann, G.Cornell, „*Java. Podstawy. Wydanie IX*", Helion, 2013
ISBN: 8324677615, 9788324677610

[3] C.E.deOliveira, G.L.Turnquist, A.Antonov, „*Developing Java Applications with Spring and Spring Boot: Practical Spring and Spring Boot solutions for building effective applications*", Packt Publishing Ltd, 2018
ISBN: 1789539137, 9781789539134

[4] R.O.Obe, L.S.Hsu, „*PostgreSQL: Up and Running*", O'Reilly Media, Inc., 2012
ISBN: 1449326293, 9781449326296

[5] S.Kunjumohamed, H.Sattari, A.Bretet, G.Warin, „*Spring MVC: Designing Real-World Web Applications*", Packt Publishing Ltd, 2016
ISBN: 1787125084, 9781787125087

[6] N.Leonard, A.Way, F.Santune, „*Web and Digital for Graphic Designers*", Bloomsbury Publishing, 2020
ISBN: 1350027561, 9781350027565

8. Spis rysunków

1. Źródło: Wykonanie własne. Zrzut ekranu ze strony domowej aplikacji. 6
2. Źródło: Wygenerowane za pomocą strony „hatchful.shopify.com". Logo strony. Prawa autorskie zezwalają na nieodpłatne i legalne kopiowanie oraz udostępnianie. 7
3. Źródło: Wykonanie własne. Diagram relacji tabel w bazie danych. 9
4. Źródło: Wykonanie własne. Zrzut ekranu ze strony z odpowiedzią algorytmu. 13