PAP2023Z-Z17 – ETAP 4.

Piotr Jabłoński, Paweł Wysocki,
Bruno Sienkiewicz, Jakub Kośla

Link do gitlaba: https://gitlab-stud.elka.pw.edu.pl/papuga/pap2023z-z17

Link do filmiku: pap2023z-z17_etap_4.mp4

1. REALIZACJA ZAŁOŻEŃ

1. **Możliwość przeglądania katalogu książek, szczegółów dotyczących książek takich jak tytuł, autor, rok wydania książki, rodzaj gatunku, ilość stron, wydawnictwo, opis - 100%**
2. **Filtrowanie podglądu według nazwy, roku, autora - 100%**
3. **Zarządzanie książkami, edycja książek, usuwanie, dodawanie rekordów książek – pracownik z poziomu swojego konta może zarządzać książkami - 100%**
4. **Wyświetlanie książek: Okładka/Szczegółowy opis z detalami – użytkownik z poziomu katalogu książek ma możliwość wybrania interesującej go książki oraz sprawdzanie dodatkowych informacji o niej – ISBN, status, data dodania, szczegółowy opis – 100%**
5. **Stworzenie konta bibliotecznego, dezaktywacja lub jego edycja – użytkownik ma możliwość utworzenia konta, zmianę hasła, czy adresu zamieszkania oraz dezaktywację konta – 100%**
6. **Stworzenie konta pracowniczego, dezaktywacja lub jego edycja – 100%**
7. **Sprawdzanie stanu książek (czy i do kiedy są wypożyczone/zarezerwowane / dostępne do wypożyczenia) - 100%**
8. **Możliwe będzie zamówienie książki, tj. w przypadku, gdy książka nie będzie dostępna, możliwe będzie wyrażenie chęci wypożyczenia i dopisania się do kolejki wypożyczeń – 100%**
9. **Statystyka wypożyczeń (najczęściej wypożyczane książki w danym okresie czasu) – 100%**
10. **Będzie możliwe również sprawdzenie historii wypożyczeń konkretnych książek – 100%**
11. **Dla każdego użytkownika dostępna będzie historia wypożyczeń m. in. z możliwością sortowania wg długości wypożyczenia – 100%**
12. **Podgląd pracowników, innych placówek naszej biblioteki - 100%**

13. **Tworzenie kolejki wypożyczeń, ustawienie ogólnego limitu, konkretny limit per książka – 100%**
14. **Definiowanie kary za przekroczenie limitu oddania książki do biblioteki – 100%**
15. **Zarządzanie kontami członkowskimi (m. in. blokada w przypadku przekroczenia terminu wypożyczenia) – 100%**
16. **Możliwość uregulowania kar za przekroczenie terminu oddania książki – 100%**
17. **Możliwość przedłużenia wypożyczenia, najwyżej jeden raz, jeżeli klient do tej pory się uregulował z wypożyczeniami, i nie ma oczekujących na wypożyczenie – 100%**
18. **Podgląd wypożyczonych obecnie pozycji – 100%**
19. **Możliwość tworzenia listy pozycji, które chce się przeczytać - 100%**
20. **Możliwość oceniania wypożyczonych książek – 100%**
21. **Możliwość zgłoszenia uszkodzenia lub zgubienia książki – 100%**

2. ZMIANY WZGLĘDEM ETAPU 3.

Na tym etapie skupiliśmy się na poprawie elementów aplikacji wskazanych przez Prowadzącego podczas prezentacji etapu 3.:

1. **Rozbudowa wyjątków oraz poprawa obsługiwania dotychczasowych wyjątków – *przykład Login.java*.**
2. **Dodanie komentarzy – *przykład plik Pap.java.***
3. **Dodanie testów sprawdzających poprawność zaimplementowanych wyjątków.**

Ponadto dodaliśmy krótki filmik przedstawiający w pigułce wprowadzone przez nas zmiany.
Link od filmiku znajduje się na początku dokumentu, jak i również filmik znajduje się w naszym repozytorium.

Szczegółowe zmiany dotyczące wyjątków:

1. **BookCreatorController, BookManagerController, BranchManagerController, EmployeeAccountCreateController, UserAccountCreateController, UserAccountManageController**: Zamiast sprawdzać kod błędu jako wartość liczbową, teraz używany jest mechanizm wyjątków. W przypadku błędu, komunikat uzyskiwany z wyjątku **ConstraintViolationException** jest używany do aktualizacji statusMessage
2. **ConstraintChecker**: Zamiast zwracać kod błędu, teraz używany jest mechanizm wyjątków, co ułatwia przekazywanie konkretnych komunikatów błędów.
   Wyjątek **ConstraintViolationException** jest rzucany z odpowiednim komunikatem w zależności od rodzaju błędu wprowadzonych danych. Równolegle zmiany te zostały przetestowane w ConstraintCheckerTest
3. **W SessionFactoryMaker**: metoda getConnection zwraca połączenie do bazy danych, a jej wywołanie może generować wyjątek SQLException. Metoda jest oznaczona jako synchronized, co oznacza, że jest bezpieczna w kontekście wielowątkowości, co może być ważne w środowisku, gdzie wiele wątków próbuje uzyskać połączenie do bazy danych jednocześnie.
4. **EmployeeAccountCreateController, EmployeeLoginScreenController, UserLoginScreenController**: Zamiast bezpośredniego porównywania do stałej, teraz użyto bloku try-catch do przechwycenia wyjątku.

Komunikat błędu jest uzyskiwany z wyjątku **InvalidCredentialsException** i używany do aktualizacji operationStatus.

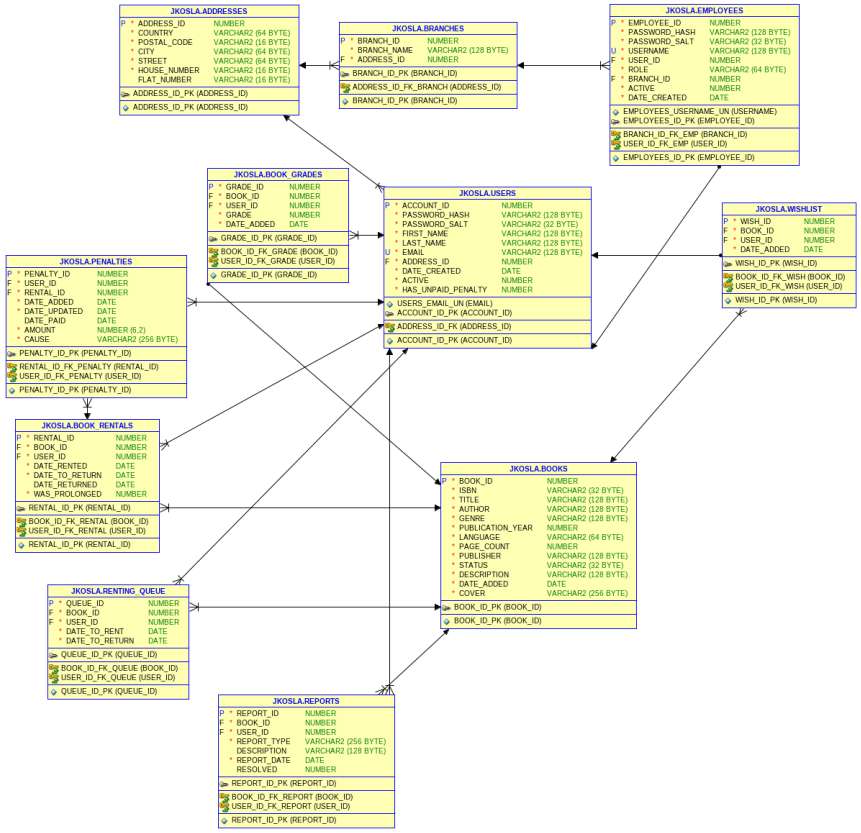
5. **Login.java**: Usunięcie klasy LoginTry: Zamiast korzystania z osobnej klasy do przechowywania kodów błędów, teraz błędy są obsługiwane poprzez rzucanie wyjątku **InvalidCredentialsException** z odpowiednimi komunikatami. Zmiana rodzaju zwracanej wartości dla metod tryLoginUser i tryLoginEmployee: Zamiast zwracać kod błędu jako wartość liczbową, teraz metody rzucają wyjątek **InvalidCredentialsException** z odpowiednim komunikatem błędu. Uzupełnienie komunikatów błędów w wyjątkach: Komunikaty w rzucanych wyjątkach zawierają teraz bardziej opisowe informacje o rodzaju błędu, co ułatwia diagnozowanie problemów. Te zmiany również zostały przetestowane w LoginTest.java

3. SCHEMAT BAZY DANYCH, WYKORZYSTANE TECHNOLOGIE

Technologie względem poprzedniego etapu nie uległy zmianie:

- **Frontend**: JavaFX
- **Backend**: Java
- **Komunikacja z bazą danych**: JPA
- **Technologia bazy danych**: PostgreSQL

Model ER bazy danych



Model fizyczny bazy danych (PDM)