**Kolegium Nauk Przyrodniczych  
Uniwersytet Rzeszowski**

**Przedmiot:**

**Bazy Danych**

**Dokumentacji Aplikacji**

**Wykonali:**

**Karol Bal**

**Jakub Bednarczyk**

**Damian Bernat**

**Filip Cyboran**

**Prowadzący:**

**Dariusz Bober**

**Rzeszów 2023**

Spis Treści

[1. Opis aplikacji i jej główny cel: 4](#_Toc136793392)

[Wstęp: 4](#_Toc136793393)

[Opis aplikacji: 4](#_Toc136793394)

[Diagram ERD: 5](#_Toc136793395)

[2. Instalacja i konfiguracja: 8](#_Toc136793396)

[1. Strona frontendowa 8](#_Toc136793397)

[2. Strona backendowa 8](#_Toc136793398)

[3. Baza danych 8](#_Toc136793399)

[3. Punkty końcowe (endpoint) API: 8](#_Toc136793400)

[1. vechicle-controller 8](#_Toc136793401)

[2. test-controller 9](#_Toc136793402)

[3. student-controller 10](#_Toc136793403)

[4. lesson-controller 11](#_Toc136793404)

[5. instructor-controller 12](#_Toc136793405)

[6. course-controller 13](#_Toc136793406)

[7. vehicle-assignment-controller 14](#_Toc136793407)

[4. Przykłady Procedur: 15](#_Toc136793408)

[Procedura get students with lessons today 15](#_Toc136793409)

[Procedura get students started in year 16](#_Toc136793410)

[Procedura get courses with lessons today 17](#_Toc136793411)

[Procedura get courses with max lessons 18](#_Toc136793412)

[Procedura get instructors with max lessons next week 19](#_Toc136793413)

[Procedura get instructors birthday in month 20](#_Toc136793414)

[Procedura get vehicles needing maintenance 21](#_Toc136793415)

[Procedura get assigned vehicles 22](#_Toc136793416)

[Procedura active assignments: 23](#_Toc136793417)

[Procedura inactive assignments 24](#_Toc136793418)

[Procedura lessons today 25](#_Toc136793419)

[Procedura upcoming lessons 26](#_Toc136793420)

[Procedura passed tests 27](#_Toc136793421)

[Procedura average test score 28](#_Toc136793422)

[5. Przykłady użycia: 29](#_Toc136793423)

[1. Rejestracja nowego studenta 29](#_Toc136793424)

[2. Przeglądanie dostępnych kursów 29](#_Toc136793425)

[3. Planowanie lekcji 29](#_Toc136793426)

[4. Przeglądanie wyników testów 29](#_Toc136793427)

[5. Zarządzanie flotą pojazdów 29](#_Toc136793428)

[6. Zabezpieczenia i ochrona danych: 30](#_Toc136793429)

[1. Strona frontendowa 30](#_Toc136793430)

[2. Strona backendowa 30](#_Toc136793431)

[3. Baza danych 30](#_Toc136793432)

# Opis aplikacji i jej główny cel:

### Wstęp:

Nasza aplikacja do zarządzania szkołą nauki jazdy została stworzona specjalnie dla szkół i ośrodków szkolenia, które oferują kursy nauki jazdy. Jest to narzędzie internetowe, które ma na celu ułatwienie zarządzania procesem nauczania i organizacją szkoły. Naszym głównym celem jest zapewnienie efektywnego i zintegrowanego rozwiązania, które umożliwia szkołom nauki jazdy sprawną obsługę uczniów, instruktorów oraz administracji. Oto kilka cech i funkcji, które są częścią naszej aplikacji do nauki jazdy:

### Opis aplikacji:

Nasza aplikacja do zarządzania szkołą nauki jazdy oferuje wiele funkcji, które usprawniają codzienne operacje szkoły i pomagają w organizacji kursów nauki jazdy. Oto kilka kluczowych cech, które można znaleźć w naszej aplikacji:

1. Zarządzanie uczniami: Aplikacja umożliwia szkole nauki jazdy skuteczne zarządzanie danymi uczniów.
2. Harmonogram lekcji: Aplikacja umożliwia zarządzanie harmonogramem lekcji.
3. Zarządzanie instruktorami: Aplikacja umożliwia szkole nauki jazdy skuteczne zarządzanie danymi instruktorów.
4. Zarządzanie pojazdami: Aplikacja umożliwia szkole nauki jazdy skuteczne zarządzanie danymi pojazdami.
5. Zarządzanie egzaminami: Aplikacja umożliwia szkole nauki jazdy skuteczne zarządzanie danymi dotyczącymi egzaminów.

Główny cel aplikacji: Głównym celem naszej aplikacji do zarządzania szkołą nauki jazdy jest usprawnienie procesu zarządzania szkołą nauki jazdy samochodem. Chcemy umożliwić szkołom skuteczne zarządzanie uczniami, instruktorami i administracją, aby cały proces był efektywny, zorganizowany i dostarczał wysoką jakość szkolenia. Nasza aplikacja ma na celu usprawnienie operacji codziennych, eliminację problemów związanych z zarządzaniem i zapewnienie pełnej kontroli nad procesem nauki jazdy.

### Diagram ERD:

Obraz zawierający tekst, zrzut ekranu, Oprogramowanie multimedialne, Oprogramowanie graficzne

Opis wygenerowany automatycznie  
Opis diagramu ERD:

Baza danych opisana powyżej składa się z kilku tabel, które służą do przechowywania informacji o kursach, instruktorach, lekcjach, studentach, testach oraz przypisaniach pojazdów.

1. Tabela "courses":
   * course\_id: unikatowy identyfikator kursu
   * course\_description: opis kursu
   * course\_duration: czas trwania kursu
   * course\_fee: opłata za kurs
   * course\_name: nazwa kursu
   * PRIMARY KEY (course\_id): klucz główny tabeli
2. Tabela "instructors":
   * instructor\_id: unikatowy identyfikator instruktora
   * address: adres instruktora
   * date\_of\_birth: data urodzenia instruktora
   * email: adres e-mail instruktora
   * first\_name: imię instruktora
   * hire\_date: data zatrudnienia instruktora
   * last\_name: nazwisko instruktora
   * phone\_number: numer telefonu instruktora
   * PRIMARY KEY (instructor\_id): klucz główny tabeli
3. Tabela "lessons":
   * lesson\_id: unikatowy identyfikator lekcji
   * end\_time: czas zakończenia lekcji
   * lesson\_date: data lekcji
   * start\_time: czas rozpoczęcia lekcji
   * course\_id: identyfikator przypisanego kursu (klucz obcy z tabeli "courses")
   * instructor\_id: identyfikator przypisanego instruktora (klucz obcy z tabeli "instructors")
   * student\_id: identyfikator ucznia (klucz obcy z tabeli "students")
   * assignment\_id: identyfikator przypisanego pojazdu (klucz obcy z tabeli "vehicle\_assignments")
   * PRIMARY KEY (lesson\_id): klucz główny tabeli
4. Tabela "students":
   * student\_id: unikatowy identyfikator studenta
   * address: adres studenta
   * date\_of\_birth: data urodzenia studenta
   * email: adres e-mail studenta
   * first\_name: imię studenta
   * last\_name: nazwisko studenta
   * phone\_number: numer telefonu studenta
   * start\_date: data rozpoczęcia kursu przez studenta
   * PRIMARY KEY (student\_id): klucz główny tabeli
5. Tabela "tests":
   * test\_id: unikatowy identyfikator testu
   * result: wynik testu (0 - niezdany, 1 - zaliczony)
   * score: wynik punktowy testu
   * test\_date: data przeprowadzenia testu
   * course\_id: identyfikator przypisanego kursu (klucz obcy z tabeli "courses")
   * instructor\_id: identyfikator przypisanego instruktora (klucz obcy z tabeli "instructors")
   * student\_id: identyfikator ucznia (klucz obcy z tabeli "students")
   * PRIMARY KEY (test\_id): klucz główny tabeli
6. Tabela "vehicle\_assignments":
   * assignment\_id: unikatowy identyfikator przypisania pojazdu (generowany automatycznie)
   * status: status przypisania pojazdu (0 - nieaktywne, 1 - aktywne)
   * vehicle\_id: identyfikator przypisanego pojazdu (klucz obcy z tabeli "vehicles")
   * PRIMARY KEY (assignment\_id): klucz główny tabeli
7. Tabela "vehicles":
   * vehicle\_id: unikatowy identyfikator pojazdu (generowany automatycznie)
   * color: kolor pojazdu
   * license\_plate: numer rejestracyjny pojazdu
   * maintenance\_date: data ostatniego przeglądu technicznego pojazdu
   * make: marka pojazdu
   * model: model pojazdu
   * registration\_date: data rejestracji pojazdu
   * year: rok produkcji pojazdu
   * PRIMARY KEY (vehicle\_id): klucz główny tabeli

Powiązania między tabelami są realizowane za pomocą kluczy obcych (foreign keys), które umożliwiają odwoływanie się do danych w innych tabelach i zapewniają spójność informacji w bazie danych.

# Instalacja i konfiguracja:

1. Strona frontendowa:
   * npm – instaluje zależnośći
   * npm run – startuje serwer aplikacji
2. Strona backendowa:
   * Należy podać hasło i nazwę bazy danych w pliku application.properties
3. Baza danych:
   * Skonfigurować bazę danych Oracle na swoim komputerze

# Punkty końcowe (endpoint) API:

1. vechicle-controller:
   * GET /api/vehicles
     + Opis: Pobiera listę wszystkich pojazdów.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę wszystkich dostępnych pojazdów w systemie.
   * POST /api/vehicles
     + Opis: Tworzy nowy pojazd.
     + Parametry żądania: Dane nowego pojazdu
     + Odpowiedź: Zwraca informacje o utworzonym pojeździe.
   * GET /api/vehicles/{vehicleId}
     + Opis: Pobiera szczegółowe informacje o pojedynczym pojeździe na podstawie jego identyfikatora.
     + Parametry żądania: vehicleId - identyfikator pojazdu.
     + Odpowiedź: Zwraca szczegółowe informacje o wybranym pojeździe.
   * DELETE /api/vehicles/{vehicleId}
     + Opis: Usuwa pojazd na podstawie jego identyfikatora.
     + Parametry żądania: vehicleId - identyfikator pojazdu.
     + Odpowiedź: Zwraca informacje potwierdzające usunięcie pojazdu.
   * GET /api/vehicles/maintance
     + Opis: Pobiera listę pojazdów wymagających przeglądu technicznego.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę pojazdów, których data ostatniego przeglądu technicznego przekracza obecny czas.
   * GET /api/vehicles/assigned
     + Opis: Pobiera listę przypisanych pojazdów.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę pojazdów, które są obecnie przypisane do instruktorów lub uczniów.
2. test-controller:
   * GET /api/tests
     + Opis: Pobiera listę wszystkich testów.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę wszystkich testów przeprowadzonych w ramach szkoły nauki jazdy.
   * POST /api/tests
     + Opis: Tworzy nowy test.
     + Parametry żądania: Dane nowego testu, takie jak identyfikator kursu, instruktora i ucznia.
     + Odpowiedź: Zwraca informacje o utworzonym teście.
   * GET /api/tests/{testsId}
     + Opis: Pobiera szczegółowe informacje o konkretnym teście na podstawie jego identyfikatora.
     + Parametry żądania: testId - identyfikator testu.
     + Odpowiedź: Zwraca szczegółowe informacje o wybranym teście
   * DELETE /api/tests/passed/{studentId}
     + Opis: Usuwa zaliczony test dla danego ucznia.
     + Parametry żądania: studentId - identyfikator ucznia.
     + Odpowiedź: Zwraca informacje potwierdzające usunięcie zaliczonego testu dla danego ucznia.
   * GET /api/tests/avarage/{courseId}
     + Opis: Pobiera średnią wyników testów dla danego kursu.
     + Parametry żądania: courseId - identyfikator kursu.
     + Odpowiedź: Zwraca średnią wyników testów dla danego kursu.
3. student-controller
   * GET /api/students
     + Opis: Pobiera listę wszystkich uczniów.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę wszystkich uczniów zarejestrowanych w systemie szkoły nauki jazdy.
   * POST /api/students
     + Opis: Tworzy nowego ucznia.
     + Parametry żądania: Dane nowego ucznia
     + Odpowiedź: Zwraca informacje o utworzonym uczniu.
   * GET /api/students/{studentId}
     + Opis: Pobiera szczegółowe informacje o konkretnym uczniu na podstawie jego identyfikatora.
     + Parametry żądania: studentId - identyfikator ucznia.
     + Odpowiedź: Zwraca szczegółowe informacje o wybranym uczniu
   * DELETE /api/students/{studentId}
     + Opis: Usuwa wybranego studenta na podstawie jego identyfikatora.
     + Parametry żądania: studentId - identyfikator studenta.
     + Odpowiedź: Zwraca informacje potwierdzające usunięcie wybranego studenta.
   * GET /api/students/todey\_lessons
     + Opis: Pobiera listę lekcji zaplanowanych na dzisiejszy dzień dla danego studenta.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę lekcji zaplanowanych na dzisiejszy dzień dla danego studenta, wraz z ich szczegółowymi danymi.
   * GET /api/students/started/{year}
     + Opis: Pobiera listę uczniów, którzy rozpoczęli naukę w określonym roku.
     + Parametry żądania: year - rok rozpoczęcia nauki.
     + Odpowiedź: Zwraca listę uczniów, którzy rozpoczęli naukę w określonym roku, wraz z ich szczegółowymi danymi.
4. lesson-controller
   * GET /api/lessons
     + Opis: Pobiera listę wszystkich lekcji.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę wszystkich lekcji przeprowadzonych w ramach szkoły nauki jazdy.
   * POST /api/lessons
     + Opis: Tworzy nową lekcję.
     + Parametry żądania: Dane nowej lekcji, takie jak data, godzina rozpoczęcia i zakończenia, identyfikator kursu, instruktora i ucznia.
     + Odpowiedź: Zwraca informacje o utworzonej lekcji.
   * GET /api/lessons/{lessonId}
     + Opis: Pobiera szczegółowe informacje o konkretniej lekcji na podstawie jej identyfikatora.
     + Parametry żądania: lessonId - identyfikator lekcji.
     + Odpowiedź: Zwraca szczegółowe informacje o wybranej lekcji, takie jak data, godzina rozpoczęcia i zakończenia, kurs, instruktor i uczeń.
   * DELETE /api/lessons/{lessonId}
     + Opis: Usuwa wybraną lekcję na podstawie jej identyfikatora.
     + Parametry żądania: lessonId - identyfikator lekcji.
     + Odpowiedź: Zwraca informacje potwierdzające usunięcie wybranej lekcji.
   * GET /api/lessons/upcoming/{instructorId}
     + Opis: Pobiera listę nadchodzących lekcji dla konkretnego instruktora.
     + Parametry żądania: instructorId - identyfikator instruktora.
     + Odpowiedź: Zwraca listę nadchodzących lekcji przypisanych do danego instruktora, wraz z ich szczegółowymi danymi.
   * Get /api/lessons/today
     + Opis: Pobiera listę lekcji zaplanowanych na dzisiejszy dzień.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę lekcji zaplanowanych na dzisiejszy dzień, wraz z ich szczegółowymi danymi.
5. instructor-controller
   * GET /api/instructors
     + Opis: Pobiera listę wszystkich instruktorów.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę wszystkich instruktorów pracujących w szkole nauki jazdy.
   * POST /api/instructors
     + Opis: Tworzy nowego instruktora.
     + Parametry żądania: Dane nowego instruktora
     + Odpowiedź: Zwraca informacje o utworzonym instruktorze.
   * GET /api/instructors/{instructorId}
     + Opis: Pobiera szczegółowe informacje o konkretnym instruktorze na podstawie jego identyfikatora.
     + Parametry żądania: instructorId - identyfikator instruktora.
     + Odpowiedź: Zwraca szczegółowe informacje o wybranym instruktorze
   * DELETE /api/instructors/{instructorId}
     + Opis: Usuwa wybranego instruktora na podstawie jego identyfikatora.
     + Parametry żądania: instructorId - identyfikator instruktora.
     + Odpowiedź: Zwraca informacje potwierdzające usunięcie wybranego instruktora.
   * GET /api/ instructors /max\_lessons\_next\_week
     + Opis: Pobiera instruktora z największą liczbą zaplanowanych lekcji na najbliższy tydzień.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca informacje o instruktorze, który ma największą liczbę zaplanowanych lekcji w najbliższym tygodniu.
   * GET /api/instructors/birthday/{month}
     + Opis: Pobiera listę instruktorów urodzonych w określonym miesiącu.
     + Parametry żądania: month - miesiąc urodzenia (np. "01" dla stycznia).
     + Odpowiedź: Zwraca listę instruktorów urodzonych w określonym miesiącu, wraz z ich szczegółowymi danymi.
6. course-controller
   * GET /api/courses
     + Opis: Pobiera listę wszystkich kursów.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę wszystkich dostępnych kursów w szkole nauki jazdy.
   * POST /api/courses
     + Opis: Tworzy nowy kurs.
     + Parametry żądania: Dane nowego kursu
     + Odpowiedź: Zwraca informacje o utworzonym kursie.
   * GET /api/courses/{courseId}
     + Opis: Pobiera szczegółowe informacje o konkretnym kursie na podstawie jego identyfikatora.
     + Parametry żądania: courseId - identyfikator kursu.
     + Odpowiedź: Zwraca szczegółowe informacje o wybranym kursie
   * DELETE /api/courses/{courseId}
     + Opis: Usuwa wybrany kurs na podstawie jego identyfikatora.
     + Parametry żądania: courseId - identyfikator kursu.
     + Odpowiedź: Zwraca informacje potwierdzające usunięcie wybranego kursu.
   * GET /api/courses/today\_lessons
     + Opis: Pobiera listę lekcji zaplanowanych na dzisiejszy dzień dla wszystkich kursów.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę lekcji zaplanowanych na dzisiejszy dzień dla wszystkich kursów, wraz z ich szczegółowymi danymi.
   * GET /api/courses/max\_lessons
     + Opis: Pobiera kurs z największą liczbą lekcji.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca informacje o kursie, który ma największą liczbę lekcji w ramach szkoły nauki jazdy.
7. vehicle-assignment-controller
   * GET /api/assignments
     + Opis: Pobiera listę wszystkich przypisań pojazdów.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę wszystkich przypisań pojazdów w szkolenauki jazdy.
   * POST /api/assignments
     + Opis: Tworzy nowe przypisanie pojazdu.
     + Parametry żądania: Dane nowego przypisania pojazdu.
     + Odpowiedź: Zwraca informacje o utworzonym przypisaniu pojazdu.
   * GET /api/assignments/{assignmentId}
     + Opis: Pobiera szczegółowe informacje o konkretnym przypisaniu pojazdu na podstawie jego identyfikatora.
     + Parametry żądania: assignmentId - identyfikator przypisania pojazdu.
     + Odpowiedź: Zwraca szczegółowe informacje o wybranym przypisaniu pojazdu.
   * DELETE /api/assignments/{assignmentId}
     + Opis: Usuwa wybrane przypisanie pojazdu na podstawie jego identyfikatora.
     + Parametry żądania: assignmentId - identyfikator przypisania pojazdu.
     + Odpowiedź: Zwraca informacje potwierdzające usunięcie wybranego przypisania pojazdu.
   * GET /api/assignments/inactive
     + Opis: Pobiera listę nieaktywnych przypisań pojazdów.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę przypisań pojazdów, które są obecnie nieaktywne.
   * GET /api/assigments/active
     + Opis: Pobiera listę aktywnych przypisań pojazdów.
     + Parametry żądania: Brak.
     + Odpowiedź: Zwraca listę przypisań pojazdów, które są obecnie aktywne.

# Przykłady Procedur:

Procedura get students with lessons today:

create or replace NONEDITIONABLE PROCEDURE get\_students\_with\_lessons\_today (students\_cursor OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN students\_cursor FOR

SELECT s.\*

FROM students s

INNER JOIN lessons l

ON s.student\_id = l.student\_id

WHERE TRUNC(l.lesson\_date) = TRUNC(SYSDATE);

END get\_students\_with\_lessons\_today;

Procedura "get\_students\_with\_lessons\_today" jest procedurą bazodanową, która służy do pobrania listy studentów, którzy mają lekcje zaplanowane na dzisiejszy dzień. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który można odczytać w celu uzyskania szczegółowych informacji o studentach.

Opis ogólny procedury:

* Nazwa procedury: get\_students\_with\_lessons\_today
* Parametry:
  + students\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które łączy tabelę studentów (students) z tabelą lekcji (lessons) na podstawie identyfikatora studenta (student\_id).
  + Następnie, za pomocą warunku WHERE, wybierane są tylko te lekcje, których data (lesson\_date) jest równa dzisiejszej dacie (SYSDATE).
  + Wynik zapytania, czyli listę studentów, którzy mają lekcje zaplanowane na dzisiaj, zapisuje się w kursorze students\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o studentach.

Procedura get students started in year:

create or replace NONEDITIONABLE PROCEDURE get\_students\_started\_in\_year (students\_cursor OUT SYS\_REFCURSOR, p\_year IN NUMBER)

IS

BEGIN

OPEN students\_cursor FOR

SELECT s.\*

FROM students s

WHERE EXTRACT(YEAR FROM s.start\_date) = p\_year;

END get\_students\_started\_in\_year;

Procedura "get\_students\_started\_in\_year" jest procedurą bazodanową, która służy do pobrania listy studentów rozpoczynających naukę w określonym roku. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który można odczytać w celu uzyskania szczegółowych informacji o studentach.

Opis ogólny procedury:

* Nazwa procedury: get\_students\_started\_in\_year
* Parametry:
  + students\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
  + p\_year (IN): Parametr wejściowy typu NUMBER, określający rok, dla którego mają być pobrani studenci.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które wybiera wszystkich studentów z tabeli students.
  + Za pomocą funkcji EXTRACT(YEAR FROM s.start\_date) wyodrębnia się rok z daty rozpoczęcia nauki studenta.
  + Warunek WHERE sprawdza, czy wyekstrahowany rok jest równy wartości przekazanej do parametru p\_year.
  + Wynik zapytania, czyli listę studentów, którzy rozpoczęli naukę w określonym roku, zapisuje się w kursorze students\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o studentach.

Procedura get courses with lessons today:

CREATE OR REPLACE PROCEDURE get\_courses\_with\_lessons\_today (courses\_cursor OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN courses\_cursor FOR

SELECT DISTINCT c.\*

FROM courses c

INNER JOIN lessons l ON l.course\_id = c.course\_id

WHERE TRUNC(l.lesson\_date) = TRUNC(SYSDATE);

END get\_courses\_with\_lessons\_today;

Procedura "get\_courses\_with\_lessons\_today" jest procedurą bazodanową, która służy do pobrania listy kursów, które mają lekcje zaplanowane na dzisiejszy dzień. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który można odczytać w celu uzyskania szczegółowych informacji o kursach.

Opis ogólny procedury:

* Nazwa procedury: get\_courses\_with\_lessons\_today
* Parametry:
  + courses\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które łączy tabelę kursów (courses) z tabelą lekcji (lessons) na podstawie identyfikatora kursu (course\_id).
  + Następnie, za pomocą warunku WHERE, wybierane są tylko te lekcje, których data (lesson\_date) jest równa dzisiejszej dacie (SYSDATE).
  + Wynik zapytania, czyli listę kursów, które mają lekcje zaplanowane na dzisiaj, zapisuje się w kursorze courses\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o kursach.

Procedura get courses with max lessons:

CREATE OR REPLACE PROCEDURE get\_courses\_with\_max\_lessons (courses\_cursor OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN courses\_cursor FOR

SELECT c.\*, COUNT(l.lesson\_id) AS lesson\_count

FROM courses c

INNER JOIN lessons l ON l.course\_id = c.course\_id

GROUP BY c.course\_id, c.course\_name, c.course\_description, c.course\_duration, c.course\_fee

ORDER BY lesson\_count DESC;

END get\_courses\_with\_max\_lessons;

Procedura "get\_courses\_with\_max\_lessons" jest procedurą bazodanową, która służy do pobrania listy kursów wraz z liczbą lekcji, posortowaną malejąco według liczby lekcji. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który można odczytać w celu uzyskania szczegółowych informacji o kursach.

Opis ogólny procedury:

* Nazwa procedury: get\_courses\_with\_max\_lessons
* Parametry:
  + courses\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które łączy tabelę kursów (courses) z tabelą lekcji (lessons) na podstawie identyfikatora kursu (course\_id).
  + Wykorzystując funkcję COUNT, obliczana jest liczba lekcji (lesson\_count) dla każdego kursu.
  + Następnie wynik grupowany jest według identyfikatora kursu oraz innych atrybutów kursu, takich jak nazwa, opis, czas trwania i opłata za kurs.
  + Ostatecznie wynik sortowany jest malejąco według liczby lekcji, aby na początku znajdowały się kursy z największą liczbą lekcji.
  + Wynik zapytania, czyli listę kursów wraz z liczbą lekcji, zapisuje się w kursorze courses\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o kursach.

Procedura get instructors with max lessons next week:

CREATE OR REPLACE PROCEDURE get\_instructors\_with\_max\_lessons\_next\_week (instructors\_cursor OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN instructors\_cursor FOR

SELECT i.\*, COUNT(l.lesson\_id) AS lesson\_count

FROM instructors i

INNER JOIN lessons l ON l.instructor\_id = i.instructor\_id

WHERE l.lesson\_date BETWEEN SYSDATE AND SYSDATE + 7

GROUP BY i.instructor\_id, i.first\_name, i.last\_name, i.date\_of\_birth, i.phone\_number, i.email, i.address, i.hire\_date

ORDER BY lesson\_count DESC;

END get\_instructors\_with\_max\_lessons\_next\_week;

Procedura "get\_instructors\_with\_max\_lessons\_next\_week" jest procedurą bazodanową, która służy do pobrania listy instruktorów wraz z liczbą lekcji, które mają zaplanowane na nadchodzący tydzień. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który można odczytać w celu uzyskania szczegółowych informacji o instruktorach.

Opis ogólny procedury:

* Nazwa procedury: get\_instructors\_with\_max\_lessons\_next\_week
* Parametry:
  + instructors\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które łączy tabelę instruktorów (instructors) z tabelą lekcji (lessons) na podstawie identyfikatora instruktora (instructor\_id).
  + Za pomocą warunku WHERE, wybierane są tylko te lekcje, których data (lesson\_date) mieści się w zakresie od dzisiejszej daty (SYSDATE) do daty dzisiejszej plus 7 dni (SYSDATE + 7).
  + Wykorzystując funkcję COUNT, obliczana jest liczba lekcji (lesson\_count) dla każdego instruktora.
  + Następnie wynik grupowany jest według identyfikatora instruktora oraz innych atrybutów instruktora, takich jak imię, nazwisko, data urodzenia, numer telefonu, adres, data zatrudnienia itp.
  + Ostatecznie wynik sortowany jest malejąco według liczby lekcji, aby na początku znajdowali się instruktorzy z największą liczbą lekcji na nadchodzący tydzień.
  + Wynik zapytania, czyli listę instruktorów wraz z liczbą lekcji, zapisuje się w kursorze instructors\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o instruktorach

Procedura get instructors birthday in month:

CREATE OR REPLACE PROCEDURE get\_instructors\_birthday\_in\_month (in\_month IN NUMBER, instructors\_cursor OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN instructors\_cursor FOR

SELECT \*

FROM instructors

WHERE TO\_CHAR(date\_of\_birth, 'MM') = TO\_CHAR(in\_month);

END get\_instructors\_birthday\_in\_month;

Procedura "get\_instructors\_birthday\_in\_month" jest procedurą bazodanową, która służy do pobrania listy instruktorów, których urodziny przypadać będą w określonym miesiącu. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który można odczytać w celu uzyskania szczegółowych informacji o instruktorach.

Opis ogólny procedury:

* Nazwa procedury: get\_instructors\_birthday\_in\_month
* Parametry:
  + in\_month (IN): Parametr wejściowy typu NUMBER, który określa numer miesiąca, dla którego mają zostać pobrani instruktorzy.
  + instructors\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które wybiera wszystkich instruktorów z tabeli instructors.
  + Za pomocą warunku WHERE, porównuje się miesiąc urodzin (date\_of\_birth) instruktorów z wartością przekazaną jako parametr in\_month.
  + Wykorzystując funkcję TO\_CHAR, konwertuje się miesiąc urodzin na format 'MM', aby porównać go z wartością in\_month.
  + Wynik zapytania, czyli instruktorów, których urodziny przypadać będą w określonym miesiącu, zapisuje się w kursorze instructors\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o instruktorach.

Procedura get vehicles needing maintenance:

CREATE OR REPLACE PROCEDURE get\_vehicles\_needing\_maintenance (vehicles\_cursor OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN vehicles\_cursor FOR

SELECT \*

FROM vehicles

WHERE maintenance\_date <= SYSDATE + 30;

END get\_vehicles\_needing\_maintenance;

Procedura "get\_vehicles\_needing\_maintenance" jest procedurą bazodanową, która służy do pobrania listy pojazdów wymagających konserwacji w określonym przedziale czasu. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który można odczytać w celu uzyskania szczegółowych informacji o pojazdach.

Opis ogólny procedury:

* Nazwa procedury: get\_vehicles\_needing\_maintenance
* Parametry:
  + vehicles\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które wybiera wszystkie pojazdy z tabeli vehicles.
  + Za pomocą warunku WHERE, sprawdza się czy data konserwacji (maintenance\_date) dla danego pojazdu jest wcześniejsza lub równa dacie bieżącej (SYSDATE) z dodanym okresem 30 dni.
  + Wynik zapytania, czyli pojazdy wymagające konserwacji w określonym przedziale czasu, zapisuje się w kursorze vehicles\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o pojazdach.

Procedura get assigned vehicles:

CREATE OR REPLACE PROCEDURE get\_assigned\_vehicles (vehicles\_cursor OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN vehicles\_cursor FOR

SELECT v.\*

FROM vehicles v JOIN vehicle\_assignments va ON v.vehicle\_id = va.vehicle\_id

WHERE va.status = 1;

END get\_assigned\_vehicles;

Procedura "get\_assigned\_vehicles" jest procedurą bazodanową, która służy do pobierania listy przypisanych pojazdów. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który można odczytać w celu uzyskania szczegółowych informacji o przypisanych pojazdach.

Opis ogólny procedury:

* Nazwa procedury: get\_assigned\_vehicles
* Parametry:
  + vehicles\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które łączy tabelę vehicles z tabelą vehicle\_assignments przy pomocy klucza obcego vehicle\_id.
  + Za pomocą warunku WHERE, sprawdza się czy status przypisania pojazdu (status) wynosi 1, co oznacza przypisanie aktywne.
  + Wynik zapytania, czyli pojazdy z aktywnym przypisaniem, zapisuje się w kursorze vehicles\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o przypisanych pojazdach.

### Procedura active assignments:

CREATE OR REPLACE PROCEDURE active\_assignments (assignments\_cursor OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN assignments\_cursor FOR

SELECT \*

FROM vehicle\_assignments

WHERE status = 1;

END active\_assignments;

Procedura "active\_assignments" jest procedurą bazodanową, która służy do pobierania aktywnych przypisań pojazdów. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który zawiera informacje o aktywnych przypisaniach pojazdów.

Opis ogólny procedury:

* Nazwa procedury: active\_assignments
* Parametry:
  + assignments\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które pobiera wszystkie rekordy z tabeli vehicle\_assignments.
  + Za pomocą warunku WHERE, sprawdza się czy status przypisania pojazdu (status) wynosi 1, co oznacza aktywne przypisanie.
  + Wynik zapytania, czyli aktywne przypisania pojazdów, zapisuje się w kursorze assignments\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o aktywnych przypisaniach pojazdów.

Procedura inactive assignments:

CREATE OR REPLACE PROCEDURE inactive\_assignments (assignments\_cursor OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN assignments\_cursor FOR

SELECT \*

FROM vehicle\_assignments

WHERE status = 0;

END inactive\_assignments;

Procedura "inactive\_assignments" jest procedurą bazodanową, która służy do pobierania nieaktywnych przypisań pojazdów. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który zawiera informacje o nieaktywnych przypisaniach pojazdów.

Opis ogólny procedury:

* Nazwa procedury: inactive\_assignments
* Parametry:
  + assignments\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które pobiera wszystkie rekordy z tabeli vehicle\_assignments.
  + Za pomocą warunku WHERE, sprawdza się czy status przypisania pojazdu (status) wynosi 0, co oznacza nieaktywne przypisanie.
  + Wynik zapytania, czyli nieaktywne przypisania pojazdów, zapisuje się w kursorze assignments\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o nieaktywnych przypisaniach pojazdów.

Procedura lessons today:

CREATE OR REPLACE PROCEDURE lessons\_today (lessons\_cursor OUT SYS\_REFCURSOR)

IS

BEGIN

OPEN lessons\_cursor FOR

SELECT \*

FROM lessons

WHERE lesson\_date = CURRENT\_DATE;

END lessons\_today;

Procedura "lessons\_today" jest procedurą bazodanową, która służy do pobierania lekcji zaplanowanych na dzisiaj. Procedura zwraca wynik w postaci kursora SYS\_REFCURSOR, który zawiera informacje o lekcjach zaplanowanych na bieżący dzień.

Opis ogólny procedury:

* Nazwa procedury: lessons\_today
* Parametry:
  + lessons\_cursor (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które pobiera wszystkie rekordy z tabeli lessons.
  + Za pomocą warunku WHERE, sprawdza się czy data lekcji (lesson\_date) jest równa dzisiejszej dacie (CURRENT\_DATE).
  + Wynik zapytania, czyli lekcje zaplanowane na dzisiaj, zapisuje się w kursorze lessons\_cursor.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o lekcjach zaplanowanych na bieżący dzień.

Procedura upcoming lessons:

CREATE OR REPLACE PROCEDURE upcoming\_lessons (

p\_instructor\_id IN instructors.instructor\_id%TYPE,

p\_result OUT SYS\_REFCURSOR

) AS

BEGIN

OPEN p\_result FOR

SELECT \*

FROM lessons

WHERE instructor\_id = p\_instructor\_id

AND lesson\_date BETWEEN CURRENT\_DATE AND (CURRENT\_DATE + 7);

END upcoming\_lessons;

Procedura "upcoming\_lessons" jest procedurą bazodanową, która służy do pobierania nadchodzących lekcji dla danego instruktora. Procedura przyjmuje identyfikator instruktora jako parametr wejściowy i zwraca wynik w postaci kursora SYS\_REFCURSOR, który zawiera informacje o nadchodzących lekcjach dla danego instruktora.

Opis ogólny procedury:

* Nazwa procedury: upcoming\_lessons
* Parametry:
  + p\_instructor\_id (IN): Parametr wejściowy typu instructors.instructor\_id%TYPE, który zawiera identyfikator instruktora.
  + p\_result (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które pobiera wszystkie rekordy z tabeli lessons.
  + Za pomocą warunku WHERE, sprawdza się czy identyfikator instruktora (instructor\_id) jest równy wartości przekazanej jako parametr p\_instructor\_id.
  + Wykorzystując warunek BETWEEN, sprawdza się czy data lekcji (lesson\_date) mieści się w zakresie od bieżącej daty (CURRENT\_DATE) do bieżącej daty powiększonej o 7 dni.
  + Wynik zapytania, czyli nadchodzące lekcje dla danego instruktora, zapisuje się w kursorze p\_result.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o nadchodzących lekcjach dla danego instruktora.

Procedura passed tests:

CREATE OR REPLACE PROCEDURE passed\_tests (

p\_student\_id IN students.student\_id%TYPE,

p\_result OUT SYS\_REFCURSOR

) AS

BEGIN

OPEN p\_result FOR

SELECT \*

FROM tests

WHERE student\_id = p\_student\_id AND result = 1;

END passed\_tests;

Procedura "passed\_tests" jest procedurą bazodanową, która służy do pobierania zaliczonych egzaminów dla danego studenta. Procedura przyjmuje identyfikator studenta jako parametr wejściowy i zwraca wynik w postaci kursora SYS\_REFCURSOR, który zawiera informacje o zaliczonych egzaminach dla danego studenta.

Opis ogólny procedury:

* Nazwa procedury: passed\_tests
* Parametry:
  + p\_student\_id (IN): Parametr wejściowy typu students.student\_id%TYPE, który zawiera identyfikator studenta.
  + p\_result (OUT): Parametr wyjściowy typu SYS\_REFCURSOR, który przechowuje wynik zapytania.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które pobiera wszystkie rekordy z tabeli tests.
  + Za pomocą warunku WHERE, sprawdza się czy identyfikator studenta (student\_id) jest równy wartości przekazanej jako parametr p\_student\_id.
  + Dodatkowo, warunek result = 1 sprawdza, czy egzamin został zaliczony.
  + Wynik zapytania, czyli zaliczone egzaminy dla danego studenta, zapisuje się w kursorze p\_result.
* Zwracany wynik:
  + Wynik zapytania jest przechowywany w kursorze SYS\_REFCURSOR, który można odczytać, aby uzyskać szczegółowe informacje o zaliczonych egzaminach dla danego studenta.

Procedura average test score:

CREATE OR REPLACE PROCEDURE average\_test\_score (

p\_course\_id IN courses.course\_id%TYPE,

p\_result OUT NUMBER

) AS

BEGIN

SELECT AVG(score) INTO p\_result

FROM tests

WHERE course\_id = p\_course\_id;

END average\_test\_score;

Procedura "average\_test\_score" jest procedurą bazodanową, która służy do obliczania średniego wyniku testów dla określonego kursu. Procedura przyjmuje identyfikator kursu jako parametr wejściowy i zwraca średnią wartość wyników testów dla tego kursu.

Opis ogólny procedury:

* Nazwa procedury: average\_test\_score
* Parametry:
  + p\_course\_id (IN): Parametr wejściowy typu courses.course\_id%TYPE, który zawiera identyfikator kursu.
  + p\_result (OUT): Parametr wyjściowy typu NUMBER, który przechowuje obliczoną średnią wartość wyników testów.
* Opis działania:
  + Procedura wykonuje zapytanie SQL, które oblicza średnią wartość wyników testów dla określonego kursu.
  + Za pomocą warunku WHERE, sprawdza się czy identyfikator kursu (course\_id) jest równy wartości przekazanej jako parametr p\_course\_id.
  + Używając funkcji AVG(), oblicza się średnią wartość kolumny score z tabeli tests.
  + Wynik obliczenia zapisuje się w parametrze p\_result.
* Zwracany wynik:
  + Obliczona średnia wartość wyników testów dla danego kursu jest przypisywana do parametru p\_result, który można odczytać i wykorzystać w dalszej logice procedury.

# Przykłady użycia:

1. Rejestracja nowego studenta:
   * Użytkownik otwiera aplikację szkoły nauki jazdy.
   * Przechodzi do sekcji rejestracji.
   * Wprowadza swoje dane osobowe, takie jak imię, nazwisko, adres, adres e-mail i numer telefonu.
   * Wybiera preferowany kurs i datę rozpoczęcia.
2. Przeglądanie dostępnych kursów:
   * Użytkownik otwiera aplikację szkoły nauki jazdy
   * Przechodzi do sekcji "Kursy".
   * Przegląda listę dostępnych kursów, ich opisy, długość trwania i opłaty.
3. Planowanie lekcji:
   * Użytkownik otwiera aplikację szkoły nauki jazdy.
   * Przechodzi do sekcji "Lekcje".
   * Wybiera datę i godzinę, w której chciałby przeprowadzić lekcję.
   * Wybiera studenta, z którym chce odbyć lekcję.
   * System potwierdza dostępność instruktora i zasobów, takich jak samochód.
   * Lekcja jest dodawana do harmonogramu instruktora i widoczna dla studenta.
4. Przeglądanie wyników testów:
   * Użytkownik otwiera aplikację szkoły nauki jazdy.
   * Przechodzi do sekcji "Moje testy" lub "Wyniki".
   * Widzi listę przeprowadzonych testów wraz z wynikami.
   * Może przeglądać informacje o każdym teście,
   * Analizuje swoje wyniki i postępy w nauce jazdy.
5. Zarządzanie flotą pojazdów:
   * Użytkownik otwiera aplikację szkoły nauki jazdy.
   * Przechodzi do sekcji "Pojazdy".
   * Może przeglądać listę dostępnych pojazdów, ich szczegóły techniczne.
   * Dodaje nowe pojazdy do floty lub usuwa nieużywane.
   * Monitoruje status przypisania pojazdów do instruktorów i aktualizuje informacje o konserwacji.

# Zabezpieczenia i ochrona danych:

1. Strona frontendowa:
   * Walidacja danych: W celu zapobiegania atakom XSS (Cross-Site Scripting) i wstrzyknięciu złośliwego kodu, aplikacja przeprowadza odpowiednią walidację i filtrację danych wprowadzanych przez użytkowników na stronie frontendowej. Są zastosowane mechanizmy, takie jak ograniczenie długości wprowadzanych danych, sprawdzenie poprawności formatu (np. adresu e-mail) oraz wykorzystanie narzędzi do sanitizacji danych.
   * Bezpieczne przechowywanie danych lokalnych: Aplikacja używa mechanizmu przechowywania danych lokalnych po stronie klienta, takiego jak ciasteczka (cookies).
2. Strona backendowa:
   * Autoryzacja i uwierzytelnianie: Wszystkie żądania kierowane są do serwera backendowego są uwierzytelnione i autoryzowane. Stosowane są mechanizmy takie jak tokeny JWT (JSON Web Tokens).
   * Zabezpieczenie routingu i endpointów: Wszystkie endpointy API są odpowiednio zabezpieczone przed nieautoryzowanym dostępem. Stosowane są mechanizmy takie jak role użytkowników w celu zapewnienia, że tylko uprawnieni użytkownicy mają dostęp do określonych endpointów.
3. Baza danych:
   * Bezpieczne przechowywanie danych: Wszystkie dane przechowywane w bazie danych są odpowiednio zabezpieczone
   * Odpowiednie zabezpieczenie zapytań do bazy danych: Aplikacja korzysta z parametryzowanych zapytań SQL, aby zapobiegać atakom SQL Injection. Unikana jest konkatenacja ciągów tekstowych w zapytaniach SQL i zamiast tego kwykorzystywane są parametry, które są odpowiednio filtrowane i walidowane.