Temat pracy:

Aplikacja do zarządzania zadaniami do wykonania (TO DO list)

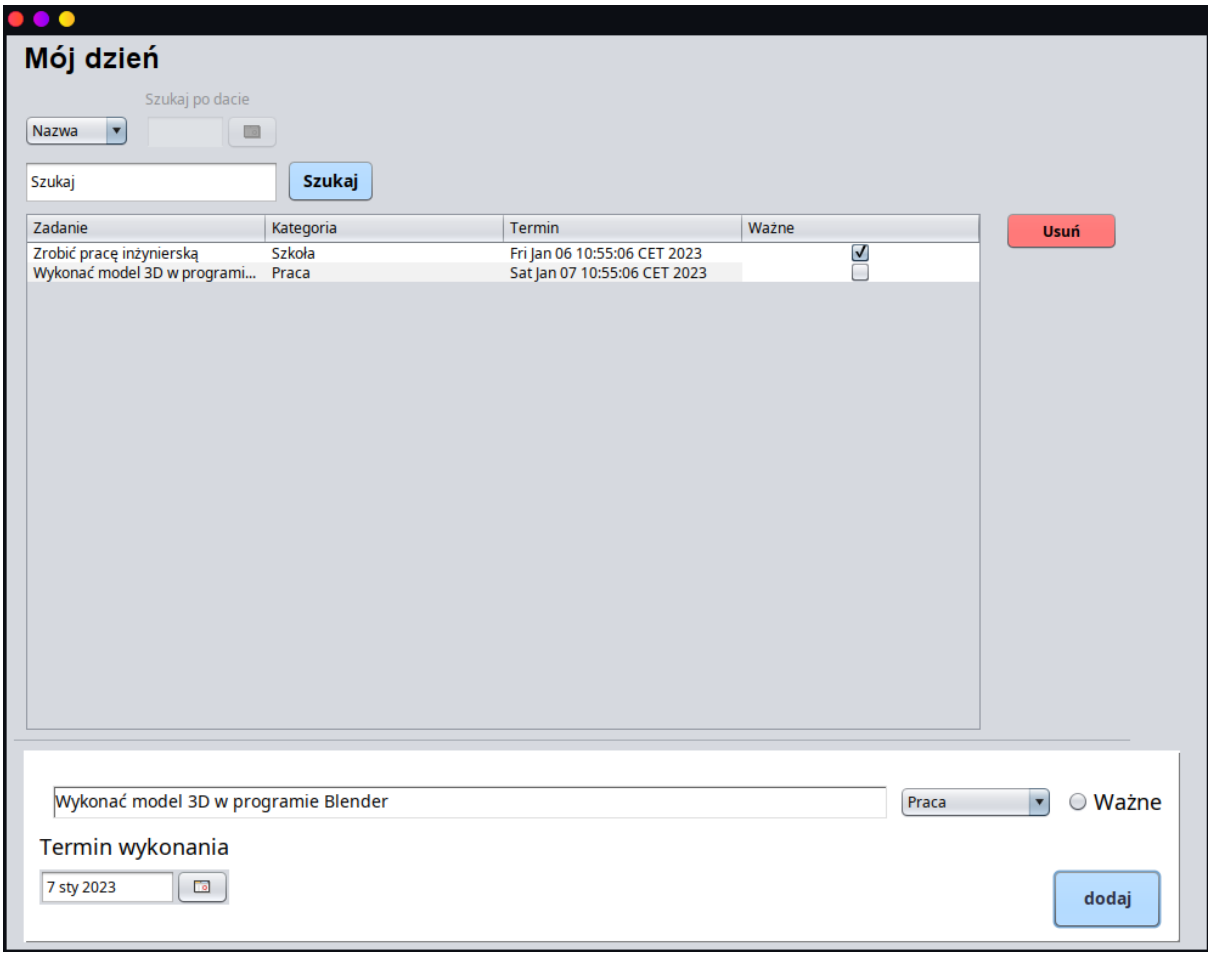
Skład zespołu:

Dariusz Knap - moduł engine
Jakub Jaroń - moduł gui
Patryk Kalita - moduł db

Opis uruchomienia
Należy przeprowadzić czystą budowę całego projektu, a następnie program należy uruchomić z modułu gui.

Moduł gui

Moduł realizujący graficzny interfejs użytkownika,który zapewnia połączenie pomiędzy użytkownikiem a funkcjami w module engine.



Rysunek przedstawiający interfejs graficzny użytkownika.

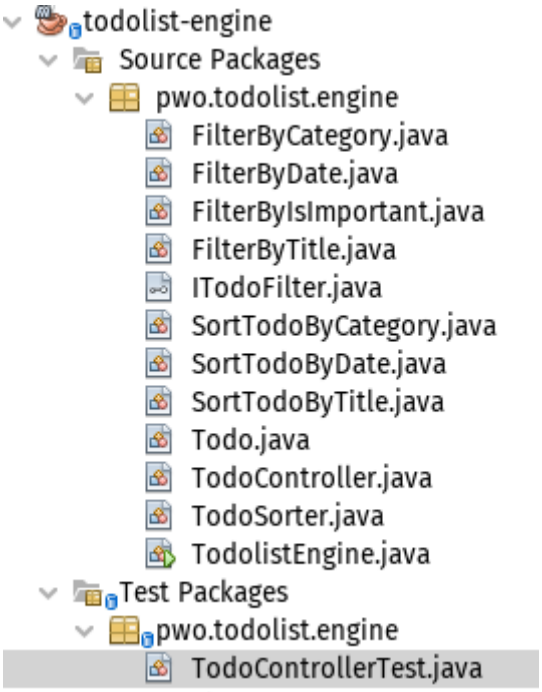
W lewym górnym rogu widnieją opcje filtrowania. Najpierw należy wybrać kolumnę po której filtrujemy, a następnie frazę lub dzień, który ma się zawrzeć w wierszach wynikowych.
Sortowanie odbywa się poprzez kliknięcie na nazwę kolumny.
By usunąć zadanie należy je zaznaczyć i kliknąć przycisk usuń.
Dolna część interfejsu odpowiedzialna jest za dodawanie zadania.

Moduł engine

Moduł zapewniający podstawową funkcjonalność systemu. Umożliwia:

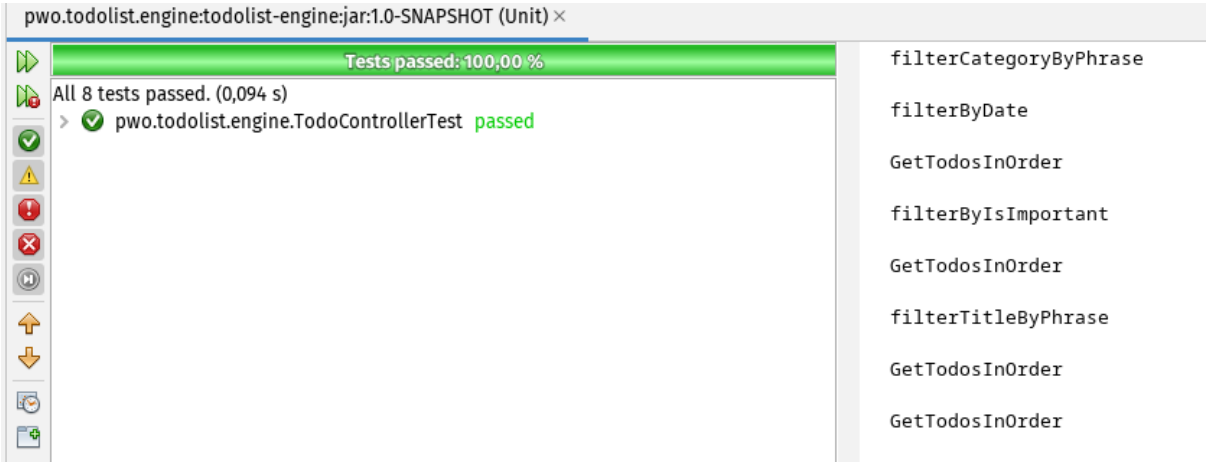
- dodawanie nowych zadań Todo,
- odczytanie wszystkich zadań,
- usunięcie zadania,
- sortowanie zadań według:
    - nazwy zadania
    - kategorii zadania
    - daty wykonania zadania
    - zaznaczonej opcji “czy ważne”
- filtrowanie zadań według:
    - frazy (fragmentu) nazwy tytułu
    - frazy kategorii zadania
    - konkretnego dnia
    - zaznaczonej opcji “czy ważne”

Sortowanie zadań jest możliwe w kolejności alfabetycznej lub w odwrotnej kolejności alfabetycznej.
Filtrowanie zadań jest niezależne od wielkości liter.



Rysunek przedstawiający strukturę modułu engine.

Do modułu stworzone zostały testy obejmujące kluczowe funkcjonalności.



Rysunek przedstawiający testy modułu engine.

```
@org.junit.jupiter.api.Test
public void testGetTodosInOrderTitleAlphabetical() {
    System.out.println("GetTodosInOrder");
    TodoSorter.SortType sortType = TodoSorter.SortType.TITLE_ALPHABETICAL;

    List<Todo> result = TodoController.getInstance().GetTodosInOrder(sortType);

    assertEquals(todos.get(2), result.get(0));
    assertEquals(todos.get(0), result.get(1));
    assertEquals(todos.get(1), result.get(2));
}
```

Rysunek przedstawiający funkcję testującą sortowanie zadań alfabetycznie po tytule

Moduł db

Moduł zapewniający zapis i odczyt danych dotyczących obiektu Todo.

- zapis wszystkich Todo
- wczytanie wszystkich Todo
- zapisanie pojedynczego Todo
- usunięcie Todo

```java
/**
 *
 * @param todo - przekazujemy obiekt todo
 * @return zapisuje obiekt todo do bazy danych
 */
public Todo AddTodo(Todo todo) {
    return todoRepository.save(todo);
}

/**
 *
 * @return wyszukuje wszystkie obiekty todo z bazy danych
 */
public List<Todo> findAllTodo() {
    return todoRepository.findAll();
}

/**
 *
 * @param todo - przekazujemny obiekt todo
 * @return aktualizujemy przekazany obiekt todo
 */
public Todo updateTodo(Todo todo) {
    return todoRepository.save(todo);
}

/**
 *
 * @param id - parametr typu int po którym szukamy obiektu todo w bazie danych
 * @return jeżeli todo istnieje zwracamy je, w przeciwwnym wypadku wyrzucony zostaje wyjątek
 */
public Todo findTodoById(Long id) {
    return todoRepository.findTodoById(id)
            .orElseThrow(() -> new TodoNotFoundException("Todo with id: " + id + " was not found."));
}

/**
 *
 * @param id - parametr typu int po którym szukamy obiektu todo w bazie danych do usunięcia
 */
public void deleteTodo(Long id) {
    todoRepository.deleteTodoById(id);
}
```

```java
public void toFile(List<Todo> todos) throws IOException {
    File csvOutputFile = new File("todos.csv");

    List<Todo> list = todos;

    //List<String> modifiedList = new ArrayList<>();

    CsvMapper mapper = new CsvMapper();
    mapper.configure(JsonGenerator.Feature.IGNORE_UNKNOWN, true);

    CsvSchema schema = CsvSchema.builder().setUseHeader(true)
            .addColumn("title")
            .addColumn("category")
            .addColumn("date")
            .addColumn("isImportant")
            .addColumn("")
            .build();

    ObjectWriter writer = mapper.writerFor(Todo.class).with(schema);

    writer.writeValues(csvOutputFile).writeAll(list);
}
```

```java
public static List<Todo> readTodosFromCSV() {

    List<Todo> todos = new ArrayList<>();
    Path pathToFile = Paths.get("todos.csv");

    try (BufferedReader br = Files.newBufferedReader(pathToFile, Charset.forName("UTF-8"))) {

        String line = br.readLine();

        while (line != null) {

            String[] attributes = line.split(",");

            Todo todo = createTodo(attributes);

            todos.add(todo);

            line = br.readLine();
        }

    } catch (IOException | ParseException ioe) {
        ioe.printStackTrace();
    }

    return todos;
}

private static Todo createTodo(String[] metadata) throws ParseException {
    String title = metadata[0];
    String category = metadata[1];
    String date = metadata[2];
    String isImportant = metadata[3];

    Date data = new SimpleDateFormat("yyyy/MM/dd").parse(date);

    boolean b1 = Boolean.parseBoolean(isImportant);

    return new Todo(title, category, data, b1);
}
```