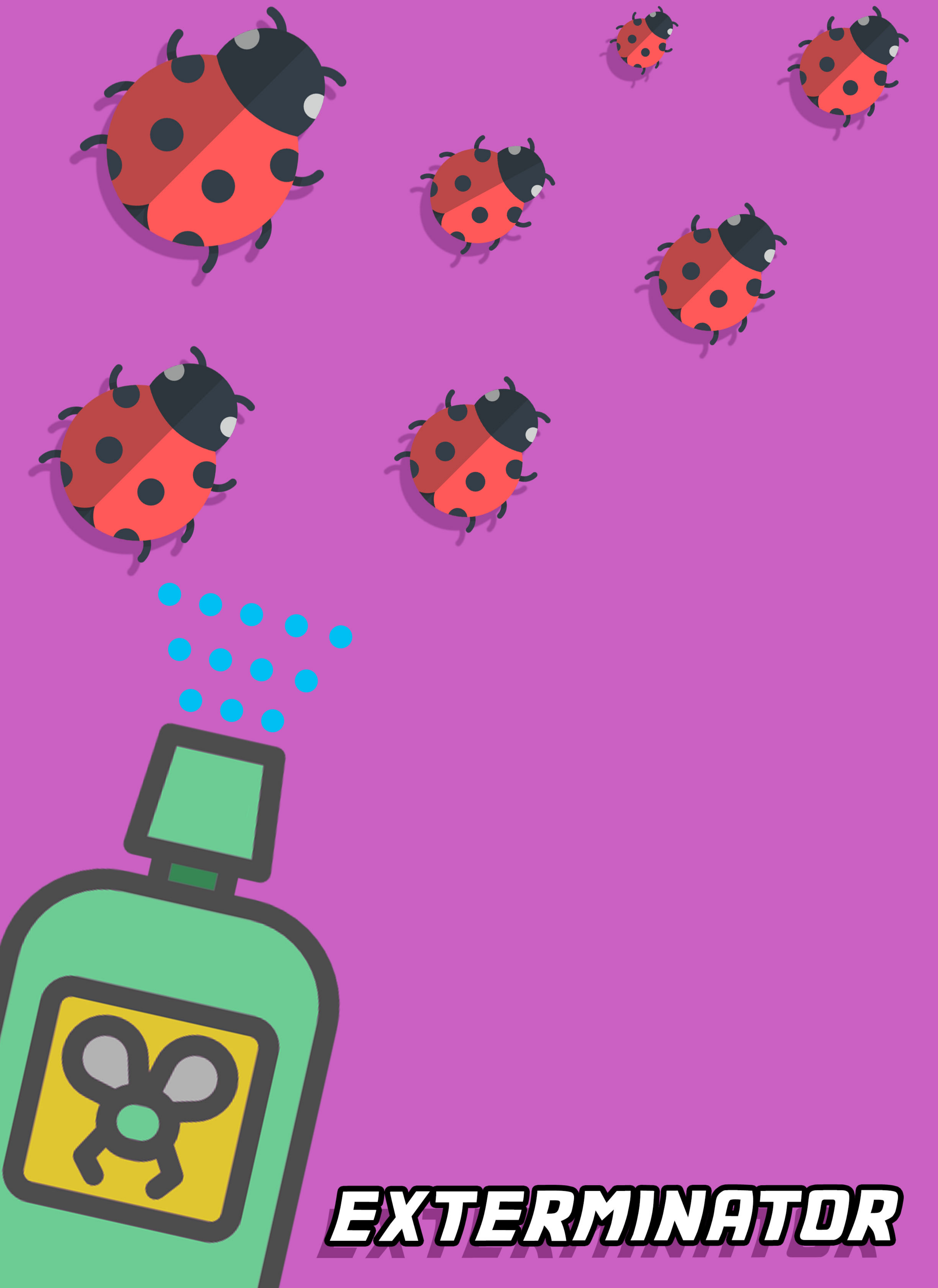
EXTERMINATOR

Small Assignment II

In this assignment we are going to create a global exception handler called **Exterminator**. Its job is to eliminate exception blindness, so we can be proactive when monitoring our applications. Let's start exterminating some bugs!

Template

There is a template already setup which can be downloaded in **Canvas** (*template.zip*). The following is included in the template:

- A three-layer structured solution with a presentation layer, service layer and repository layer
- **Exterminator.Models** includes all *InputModels, DTOs* and *Entities*
- **Exterminator.Repositories** includes:
  - **GhostbusterRepository** - *a layer on top of a static list*
  - **LogRepository** - *a layer on top of a SQLite database called LogDb.db which resides in the project*
  - **Data/** folder containing all data
- **Exterminator.Services** includes:
  - **GhostbusterService**
  - **LogService**
- **Exterminator.WebApi** includes:
  - **GhostbusterController** - *Contains three routes: GetAllGhostbusters, GetGhostbusterById and CreateGhostbuster*
  - **LogController** - *Is initially empty, but should include a single route (see below)*
  - **ExceptionHandlerExtensions** - *Contains extension methods to apply global exception handling within the application (must be implemented)*
  - **ModelStateExtensions** - *Contains extension methods for the ModelState property which can be accessed to validate the incoming model*

Assignment description

Below is a description of the functionality that should be in the web service:

- **(10%)** Register all dependencies within **Startup.cs**

  - **ILogService**
  - **IGhostbusterService**
  - **ILogRepository**
  - **IGhostbusterRepository**
  - **IGhostbusterDbContext**

- **(10%)** Create a new folder called **Exceptions/** in the **Exterminator.Models** project and create two new custom exceptions called **ResourceNotFoundException** and **ModelFormatException**. Each custom exception should inherit from **Exception**.

- **(40%)** A global exception handling should be setup within the **UseGlobalExceptionHandler** method which resides within the file **ExceptionHandlerExtensions**. The global exception handler should do the following:

  a. Retrieve information about the current exception using the **IExceptionHandlerFeature**
  b. Set a default return status code of **Internal Server Error (500)**
  c. Handle different exceptions:
     i. **ResourceNotFoundException** should return a status code **Not Found (404)**
     ii. **ModelFormatException** should return a status code **Precondition Failed (412)**
     iii. **ArgumentOutOfRangeException** should return a status code **Bad Request (400)**
  d. The response should have the **Content-Type** header set as **application/json**

e. The exception should be logged using the **ILogService.LogToDatabase()** method which accepts an **ExceptionModel** as parameter. The **ExceptionModel** should be properly filled out using information from the exception
f. The response should be written out with the exception model in string format (*JSON*)

- **(20%)** A custom validation attribute should be implemented and applied for the **GhostbusterInputModel** called **Expertize** and should do the following:

  a. It should be stored in a folder called **Attributes/** within the **Exterminator.Models** project
  b. The incoming value should only be valid if it is one of these: “Ghost catcher”, “Ghoul strangler”, “Monster encager” or “Zombie exploder”
  c. If the value is not valid, the attribute should return a descriptive error message

- **(20%)** Create a route within **LogController** called **GetAllLogs** which returns all logs using the **ILogService** which needs to be injected through the constructor. The **GetAllLogs** method is not implemented in **ILogService** nor **ILogRepository** so it’s your job to create these methods as well.

Submissions

A single compressed file (***.zip, *.rar**) should be submitted to **Canvas.**