

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Проект з дисципліни
«Інформаційні технології»

Виконав:
студент 4 курсу ОС «Бакалавр»
спеціальності «Комп'ютерні науки»,
освітньої програми «Інформатика»,
групи ТТП-42
Расахацький Максим Володимирович

Київ-2022

1. Опис завдання

Реалізувати Фрагментарну реалізацію систем управління табличними базами даних

1.1. Загальні вимоги

- кількість таблиць принципово не обмежена (реляції між таблицями не враховувати);
- кількість полів та кількість записів у кожній таблиці також принципово не обмежені.
- У кожній роботі треба забезпечити підтримку (для полів у таблицях) наступних (загальних для всіх варіантів!) типів:
 - integer;
 - real;
 - char;
 - string.
- Також у кожній роботі треба реалізувати функціональну підтримку для
 - створення бази;
 - створення (із валідацією даних) та знищення таблиці з бази;
 - перегляду та редагування рядків таблиці;
 - збереження табличної бази на диску та, навпаки, зчитування її з диску.

1.2. Варіанти додаткових типів

Потрібно забезпечити підтримку (для можливого використання у таблицях) двох додаткових типів у відповідності з одним із наступних варіантів: html-файли; stringInvl (інтервал рядків);

1.3. Варіанти додаткових операцій над таблицями

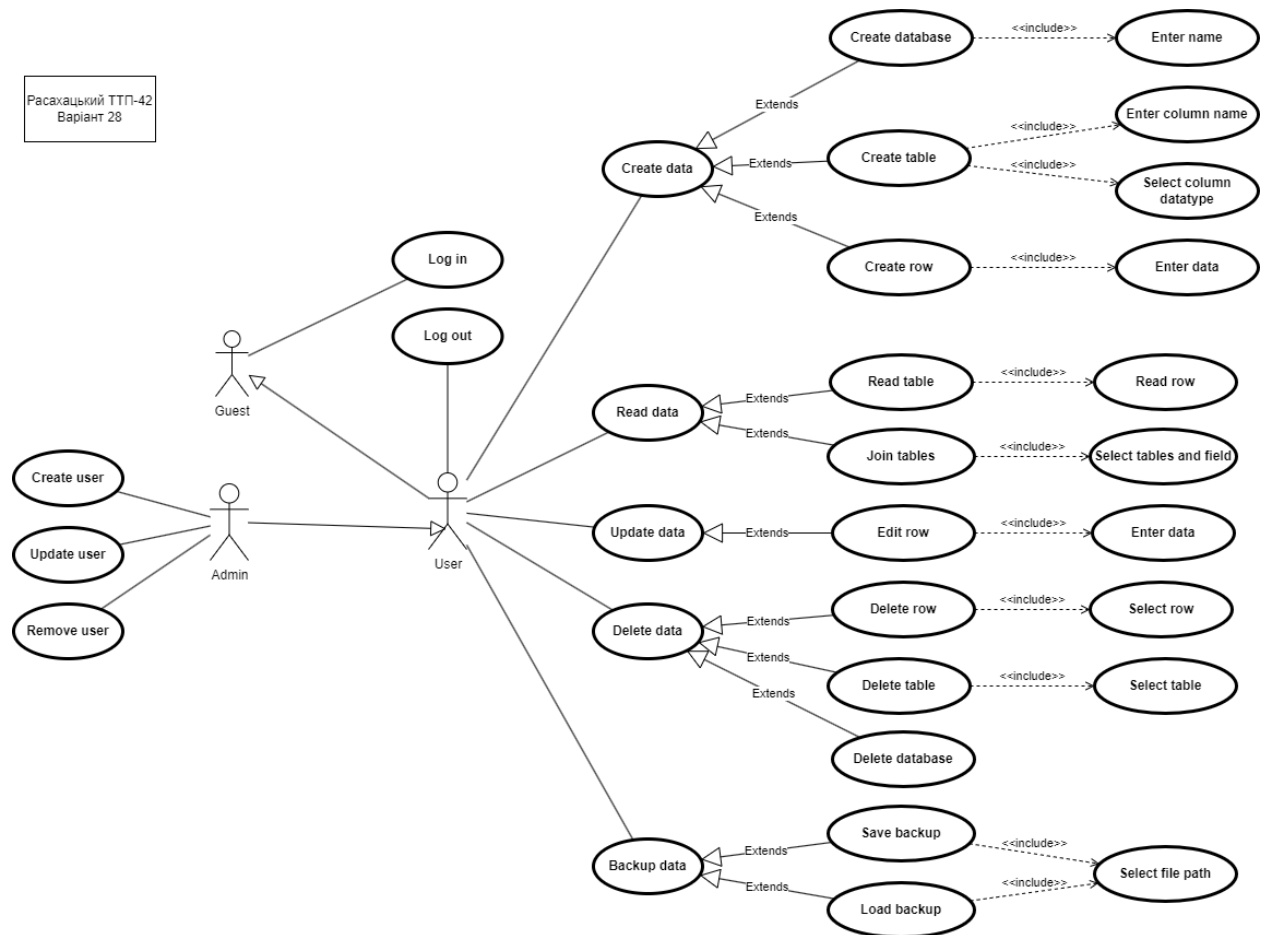
Потрібно реалізувати операції над таблицями у відповідності з варіантом 8: сполучення таблиць (за спільним полем);

2. Опис реалізації

Під час виконання лабораторних робіт було виконано наступні етапи:

2.1. Етап 0 - Попередній етап

Функціональна специфікація системи управління табличними базами даних (СУТБД) у вигляді однієї або кількох діаграм прецедентів UML.



2.2. Етап 1-2 - Розробка локальної (не розподіленої) версії СУТБД (із власною реалізацією класів "Таблиця" та "База")

Розробка власних класів для понять "Таблиця", "База" та, можливо, деяких інших класів, спряжених із поняттям "Таблиця".

Реалізовано завдання було мовою Golang, в якій немає класів, але є структури та функції що реалізують перні інтерфейси.

Створені структури:

```
type Database struct {
    Name string
    Tables []Table
}

type DBPathJSON struct {
    Name string `json:"name"`
}

type TableHeaderJSON struct {
    Name string `json:"name"`
    Type string `json:"type"`
}

type TableJSONValues struct {
    Name string `json:"name"`
    Headers []TableHeaderJSON `json:"headers"`
    Values [][]interface{} `json:"values"`
}

type TableJSON struct {
    Name string `json:"name"`
    Headers []TableHeaderJSON `json:"headers"
}
```

```
}
```

```
type DatabaseInfoJSON struct {  
    Tables []TableJSON `json:"tables"`  
}
```

```
type Table struct {  
    Name string  
    Types []string  
    Headers []string  
    Values [][]DBType  
}
```

```
type DBType interface {  
    Value() interface{}  
    TypeName() string  
}
```

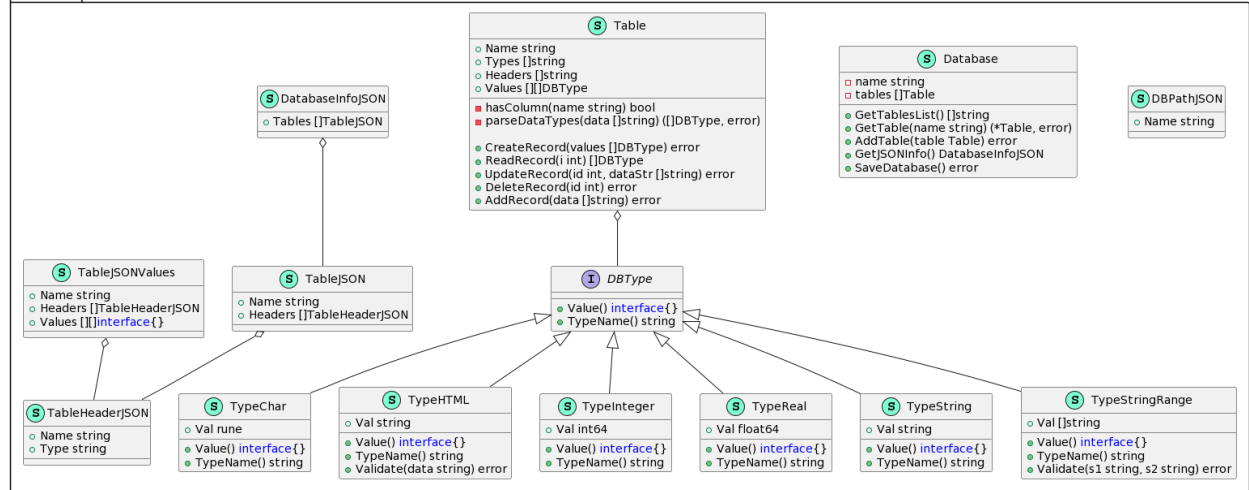
Створення UML-діаграми класів (з наявними між класами відношеннями).

Для генерації діаграми класів використано утиліту [goplantuml](#)

Для рендеру діаграми використано [dumels](#)



database



Проведення unit-тестування. Надати 3..* тести, один з яких має бути призначеним для тестуванням “індивідуальної” (варіантної) операції з розділу III.

Юніт тести знаходяться в файлі DBMS/main_test.go

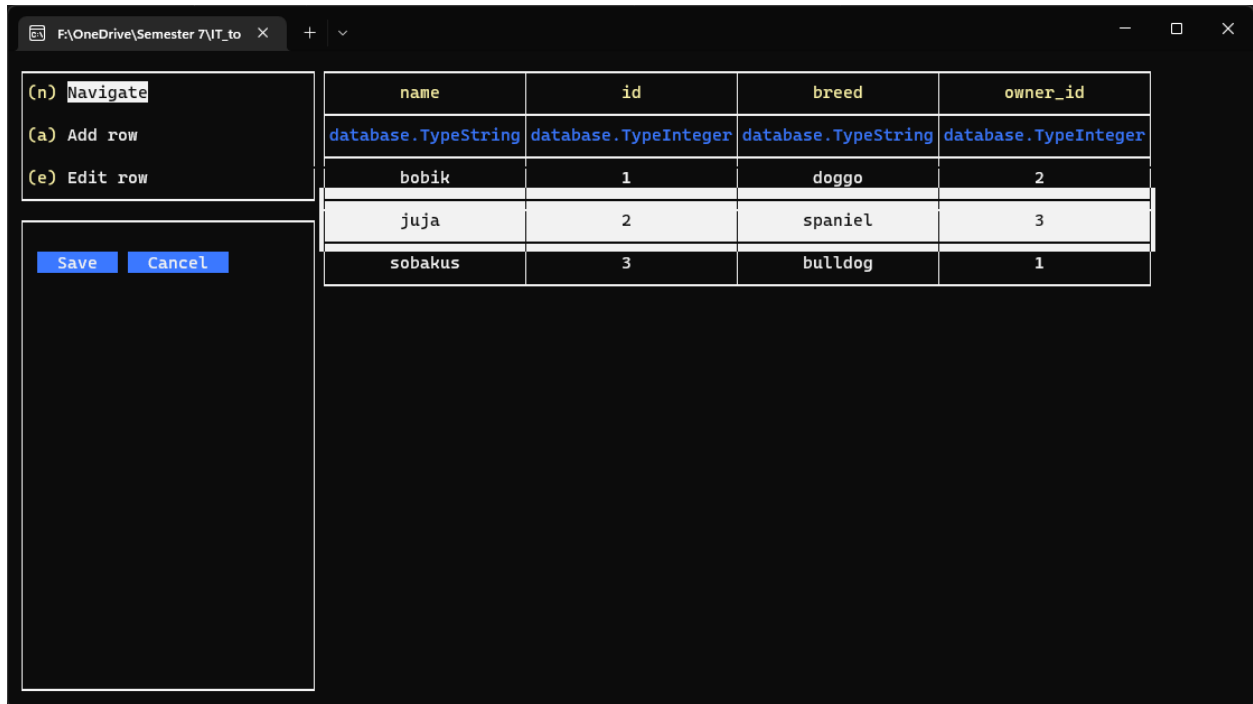
Результат виконання тестів:

```
+<4 go setup calls>
=== RUN    TestCreateDatabase
--- PASS: TestCreateDatabase (0.06s)
=== RUN    TestReadDatabasePaths
--- PASS: TestReadDatabasePaths (0.00s)
=== RUN    TestLoadDatabase
--- PASS: TestLoadDatabase (0.00s)
=== RUN    TestAddTable
--- PASS: TestAddTable (0.00s)
=== RUN    TestGetTable
--- PASS: TestGetTable (0.00s)
=== RUN    TestJoinTables
--- PASS: TestJoinTables (0.00s)
=== RUN    TestValidateStringRange_1
--- PASS: TestValidateStringRange_1 (0.00s)
=== RUN    TestValidateStringRange_2
--- PASS: TestValidateStringRange_2 (0.00s)
=== RUN    TestValidateHTML_1
--- PASS: TestValidateHTML_1 (0.00s)
=== RUN    TestValidateHTML_2
--- PASS: TestValidateHTML_2 (0.00s)
PASS

Process finished with the exit code 0
```


Забезпечення інтерфейсу користувача на основі форм.

Було використано CLI бібліотеку [tview](#)



Навігація таблицею

F:\OneDrive\Semester 7\IT_to X + ▾

(a) Add row
(e) Edit row
(d) Delete row

name:
id:
breed:
owner_id:

name	id	breed	owner_id
database.TypeString	database.TypeInteger	database.TypeString	database.TypeInteger
bobik	1	doggo	2
juja	2	spaniel	3
sobakus	3	bulldog	1

Редагування поля

F:\OneDrive\Semester 7\IT_to X + ▾

Select tables for join

Table 1:
Table 2:

Select columns from tables for join

Column from table 1:
Column from table 2:

Запуск операції Join

name	id	breed	owner_id	name	weight
database.TypeString	database.TypeInteger	database.TypeString	database.TypeInteger	database.TypeString	database.TypeRe...
bobik	1	doggo	2	maksym	69
juja	2	spaniel	3	kira	47.7
sobakus	3	bulldog	1	oleg	123

Результат операції Join

2.3. 10) REST web-сервіси. Реалізація операцій над даними, орієнтовані на їх ієрархічну структуру: база -> таблиця -> ... та на використання HTTP-запитів (як мінімум GET, POST та DELETE). Потрібно розробити REST API сервер та продемонструвати його роботу на відповідних тестових HTTP-запитах (Postman, cURL тощо).

Спочатку всі запити та інтерфейси до них були написані вручну. Коли було вивчено генерацію коду, також додатково були реалізовані генеровані інтерфейси.

Приклади запитів

GET databases list

```
GET http://localhost:1323/databases
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Vary: Origin
Date: Thu, 01 Dec 2022 22:42:09 GMT
Content-Length: 37
```

```
[
  {
    "name": "Animals"
  },
  {
    "name": "test"
  }
]
```

GET table Dogs from database Animals

```
GET http://localhost:1323/databases/Animals/Dogs
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Vary: Origin
Date: Thu, 01 Dec 2022 22:49:49 GMT
Content-Length: 295
```

```
{
  "name": "Dogs",
  "headers": [
    {
      "name": "name",
      "type": "database.TypeString"
    },
  ],
}
```

```

    {
      "name": "id",
      "type": "database.TypeInteger"
    },
    {
      "name": "breed",
      "type": "database.TypeString"
    },
    {
      "name": "owner_id",
      "type": "database.TypeInteger"
    }
  ],
  "values": [
    [
      "bobik",
      1,
      "doggo",
      2
    ],
    [
      "juja",
      2,
      "spaniel",
      3
    ],
    [
      "sobakus",
      3,
      "bulldog",
      1
    ]
  ]
}

```

GET joined table by column 'owner_id' from table 'Dogs' and 'id' from table 'owners'

```

GET
http://localhost:1323/databases/animals/joined_tables?t1=Dogs&t2=owners&c1=owner_id&c2=id

```

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Vary: Origin
Date: Thu, 01 Dec 2022 22:54:41 GMT
Content-Length: 487

```

```

{
  "name": "Dogs_join_owners",
  "headers": [
    {
      "name": "name",
      "type": "database.TypeString"
    },
  ],

```

```

    {
      "name": "id",
      "type": "database.TypeInteger"
    },
    {
      "name": "breed",
      "type": "database.TypeString"
    },
    {
      "name": "owner_id",
      "type": "database.TypeInteger"
    },
    {
      "name": "name",
      "type": "database.TypeString"
    },
    {
      "name": "weight",
      "type": "database.TypeReal"
    },
    {
      "name": "license",
      "type": "database.TypeChar"
    }
  ],
  "values": [
    [
      "bobik",
      1,
      "doggo",
      2,
      "maksym",
      69,
      82
    ],
    [
      "juja",
      2,
      "spaniel",
      3,
      "kira",
      47.7,
      81
    ],
    [
      "sobakus",
      3,
      "bulldog",
      1,
      "oleg",
      123,
      72
    ]
  ]
}

```

POST new owner

POST http://localhost:1323/databases/animals/owners/new_row

HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
Vary: Origin
Date: Thu, 01 Dec 2022 23:02:27 GMT
Content-Length: 25

```
[  
  "Andrii",  
  "4",  
  "110",  
  "G"  
]
```

DELETE 5th row in table owners

DELETE http://localhost:1323/databases/animals/owners/5

HTTP/1.1 200 OK
Content-Type: text/plain; charset=UTF-8
Vary: Origin
Date: Thu, 01 Dec 2022 23:05:36 GMT
Content-Length: 7

deleted

Response code: 200 (OK); Time: 3ms (3 ms); Content length: 7 bytes (7 B)

PUT edit owner 4

PUT http://localhost:1323/databases/animals/owners/4

HTTP/1.1 200 OK
Content-Type: text/plain; charset=UTF-8
Vary: Origin
Date: Thu, 01 Dec 2022 23:13:05 GMT
Content-Length: 8

modified

2.4. REST web-сервіси. Розробка OpenAPI Specification для взаємодії з ієрархічними даними (база, таблиця, ...).

Було розроблено OpenAPI Specification, файл DBMS/сpec.yaml

Фрагмент реалізації

```
1  openapi: 3.0.3
2  info:
3    title: My DBMS specs
4    description: I have no time hlep me
5    version: 1.0.0
6  paths:
7    "/databases":
8      get:
9        summary: Get databases list
10       tags:
11         - database
12       responses:
13         200:
14           description: Databases list
15           content:
16             application/json:
17               schema:
18                 type: array
19                 items:
20                   type: string
21           default:
22             description: Unexpected error
23             content:
24               application/json:
25                 schema:
26                   $ref: "#/components/schemas/Error"
27       post:
28         summary: Create database
29         tags:
30           - database
31         requestBody:
32           description: Database name
33           required: true
34           content:
35             application/json:
36               schema:
37                 type: string
38         responses:
39           201:
40             description: Database created
41           default:
42             description: Unexpected error
43             content:
44               application/json:
45                 schema:
46                   $ref: "#/components/schemas/Error"
```


2.5. 13) REST web-сервіси. Реалізація серверного проєкту, використовуючи кодогенерацію стабу за OpenAPI Specification.

Генерація серверного Golang коду виконана за допомогою [oapi-codegen](#)

Команда генерації

```
oapi-codegen -generate server -package genserver .\spec.yaml >
.\genserver\my.gen.go
```

Генерований код міститься в файлі /genserver/my.gen.go

Фрагмент генерованого коду

```
3
4 // ServerInterface represents all server handlers.
5 type ServerInterface interface { 4 usages  Maksym Rasakhatskiy +1
6     // Get databases list
7     // (GET /databases)
8     GetDatabases(ctx echo.Context) error
9     // Create database
10    // (POST /databases)
11    PostDatabases(ctx echo.Context) error
12    // Delete database
13    // (DELETE /databases/{db_name})
14    DeleteDatabasesDbName(ctx echo.Context, dbName string) error
15
16    // (GET /databases/{db_name})
17    GetDatabasesDbName(ctx echo.Context, dbName string) error
18    // Add new table
19    // (POST /databases/{db_name})
20    PostDatabasesDbName(ctx echo.Context, dbName string) error
```

Допоміжні типи створені вручну для передачі параметру в запиті /genserver/utills.go

```
package genserver

type GetDatabasesDbNameJoinedTablesParams struct { 4 usages  Maksym Rasakhatskiy
    T1 string
    T2 string
    C1 string
    C2 string
}
```

Імплементація серверних методів в файлі /serverImpl/impl.go

Фрагмент коду

```
60
61 func (s *MyServerImpl) GetDataBasesDbName(ctx echo.Context, dbName string) error { 1 usage  👤 Rasakhatskiy
62     db, err := database.LoadDatabase(dbName)
63     if err != nil : ctx.JSON(http.StatusNotFound, err.Error())
64
65     tables := db.GetTablesList()
66
67     return ctx.JSON(http.StatusOK, tables)
68 }
69
70
71
72 func (s *MyServerImpl) GetDataBases(ctx echo.Context) error { 1 usage  👤 Maksym Rasakhatskiy
73     response := database.ReadDatabasesPaths()
74     return ctx.JSON(http.StatusOK, response)
75 }
76
77 func (s *MyServerImpl) PostDatabases(ctx echo.Context) error { 1 usage  👤 Maksym Rasakhatskiy
78     data := new(string)
79     err := ctx.Bind(data)
80     if err != nil : ctx.String(http.StatusBadRequest, StatusBadRequest)
81     fmt.Println(*data)
82
83     err = database.CreateDatabase(*data)
84     if err != nil : ctx.String(http.StatusBadRequest, StatusBadRequest)
85
86     return ctx.JSON(http.StatusCreated, data)
87 }
88
89
90
91 }
```

2.6. 14) REST web-сервіси. Реалізація клієнтського проєкту за OpenAPI Specification.

Так як веб інтерфейс та CLI були розроблені до генерації, вони були згодом переналаштовані на генерований код та його імплементації.

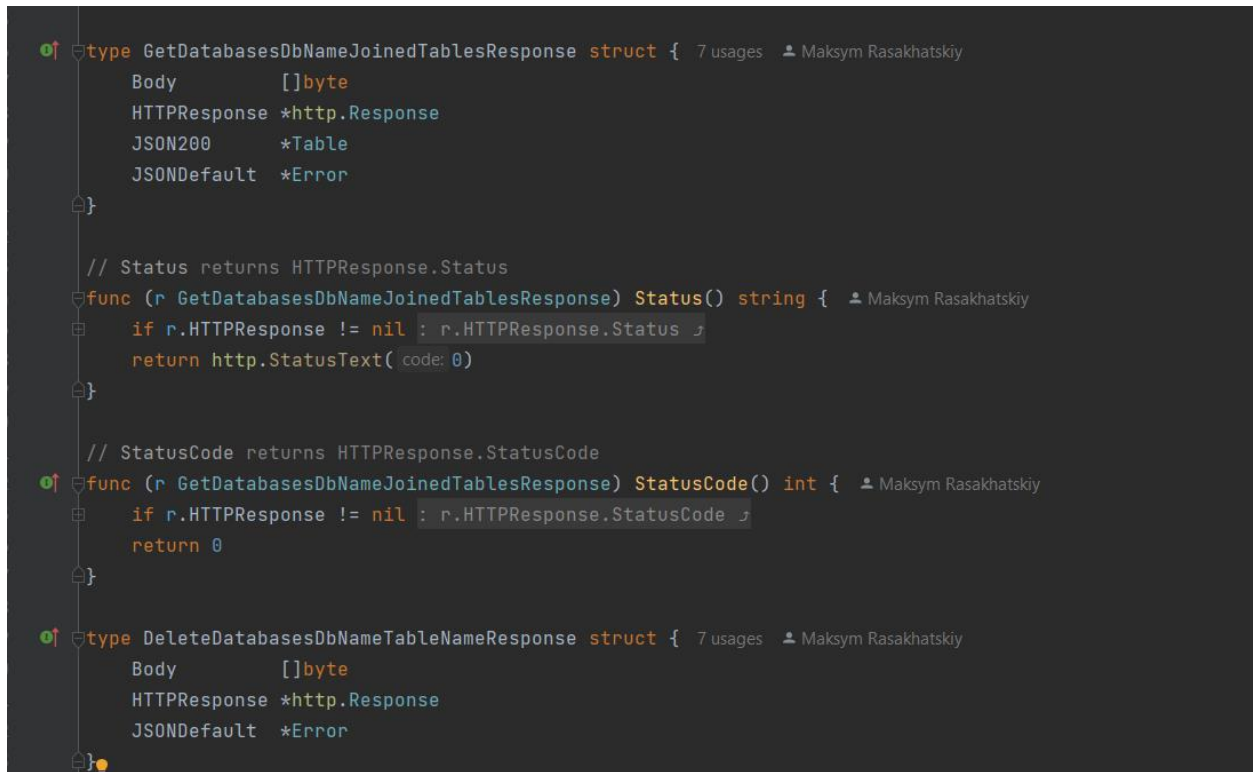
Генерація клієнтського Golang коду виконана за допомогою [oapi-codegen](#)

Команда для генерації

```
oapi-codegen -generate client -package genclient .\spec.yaml >
.\genclient\my.gen.go
```

Генерований код в файлі /genclient/my.gen.go

Фрагмент генерованого коду



```
type GetDatabasesDbNameJoinedTablesResponse struct { 7 usages  Maksym Rasakhatskiy
    Body []byte
    HTTPResponse *http.Response
    JSON200 *Table
    JSONDefault *Error
}

// Status returns HTTPResponse.Status
func (r GetDatabasesDbNameJoinedTablesResponse) Status() string {  Maksym Rasakhatskiy
    if r.HTTPResponse != nil { r.HTTPResponse.Status }
    return http.StatusText(0)
}

// StatusCode returns HTTPResponse.StatusCode
func (r GetDatabasesDbNameJoinedTablesResponse) StatusCode() int {  Maksym Rasakhatskiy
    if r.HTTPResponse != nil { r.HTTPResponse.StatusCode }
    return 0
}

type DeleteDatabasesDbNameTableNameResponse struct { 7 usages  Maksym Rasakhatskiy
    Body []byte
    HTTPResponse *http.Response
    JSONDefault *Error
}
```

Допоміжні типи створені вручну в файлі /genclient/utils.go

```
package genclient

type PostDatabasesJSONRequestBody struct { 5 usages  Maksym Rasakhatskiy *
    Name string `json:"name"`
}

type TableHeaderJSON struct { 1 usage  new *
    Name string `json:"name"`
    Type string `json:"type"`
}

// PostDatabasesDbNameJSONRequestBody table
type PostDatabasesDbNameJSONRequestBody struct { 6 usages  Maksym Rasakhatskiy *
    Name      string          `json:"name"`
    Headers []TableHeaderJSON `json:"headers"`
    Values  [][]interface{}    `json:"values"`
}

type GetDatabasesDbNameJoinedTablesParams struct { 5 usages  Maksym Rasakhatskiy
    T1 *string
    T2 *string
    C1 *string
    C2 *string
}

type Error struct { 24 usages  Maksym Rasakhatskiy *
    Code int
}
```

Імплементація клієнтських методів в файлі / genImpl/impl.go

Фрагмент коду

```
18
19 // SetupHandler creates echo server, registers handlers and starts server
20 func SetupHandler() { 1 usage  🧑 Maksym Rasakhatskiy
21     var myApi MyServerImpl
22     e := echo.New()
23     genserver.RegisterHandlers(e, &myApi)
24     e.Use(middleware.CORSWithConfig(middleware.DefaultCORSConfig))
25     e.HTTPErrorHandler = func(err error, c echo.Context) {
26         _ = c.JSON(http.StatusInternalServerError, map[string]interface{}{
27             "code": 500,
28             "message": err.Error(),
29         })
30     }
31     e.Logger.Fatal(e.Start(address: ":1323"))
32 }
33
34 func tableToJson(table *database.Table) *database.TableJSONValues { 2 usages  🧑 Maksym Rasakhatskiy
35     var headers []database.TableHeaderJSON
36     for i := range table.Headers {
37         headers = append(headers, database.TableHeaderJSON{
38             Name: table.Headers[i],
39             Type: table.Types[i],
40         })
41     }
42
43     interValues := make([][]interface{}, len(table.Values))
```

2.7. 18-19) Один або два варіанти Web-проектів. (AspNetWebApi, ASP .NET, ASP .NET MVC, WPF, JSP, JavaServlet, Spring, Struts, Struts 2, JSF, Tapestry, Wicket, GWT тощо). Кожен проект може бути функціонально обмеженим (*).

Бекенд написаний використовуючи пакет [Echo](#) для golang

Інтерфейс побудовано за допомогою [Svelte Kit](#)

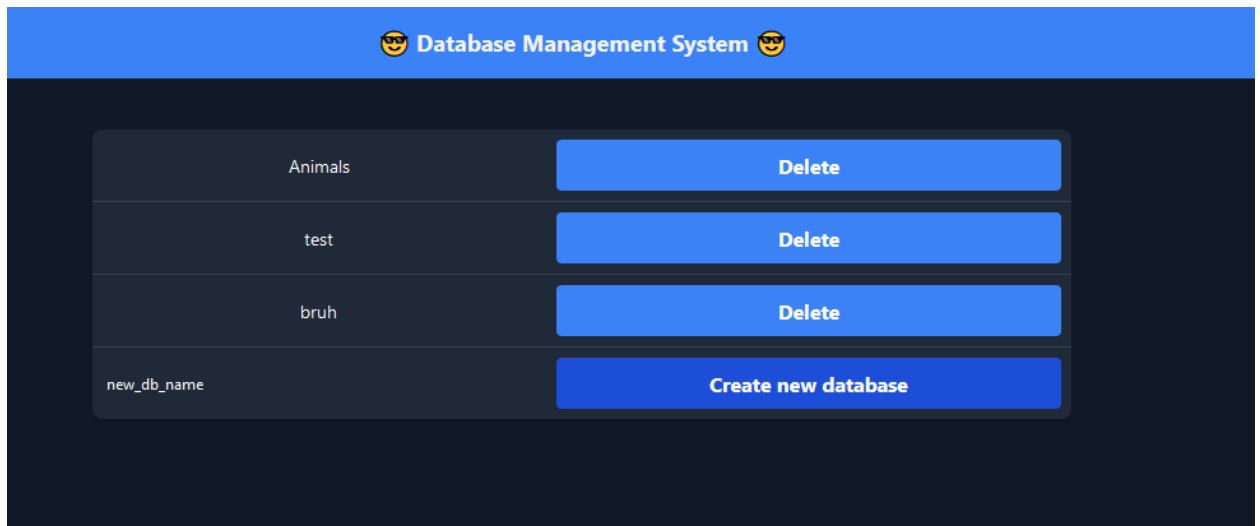
Використано компоненти [tailwind](#)

Тулінг - [Vite](#)

Список баз даних

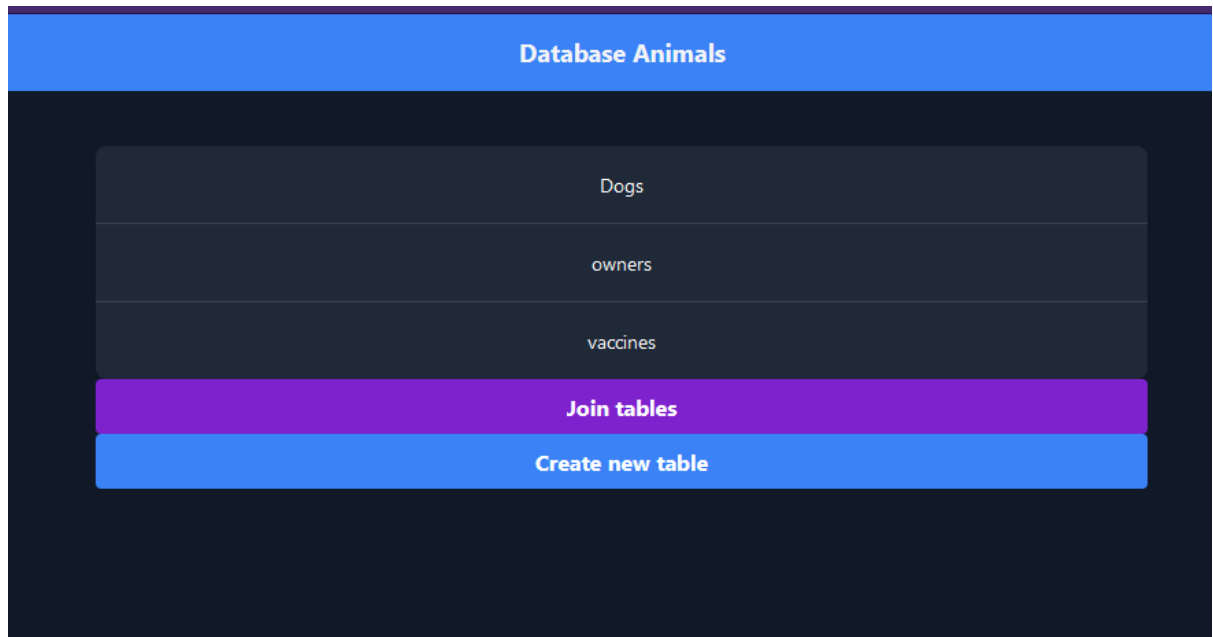
Доступні операції

- Створення бази даних
- Видалення бази даних



Список таблиц в базі даних Доступні операції

- Створити таблицю
- Join Tables

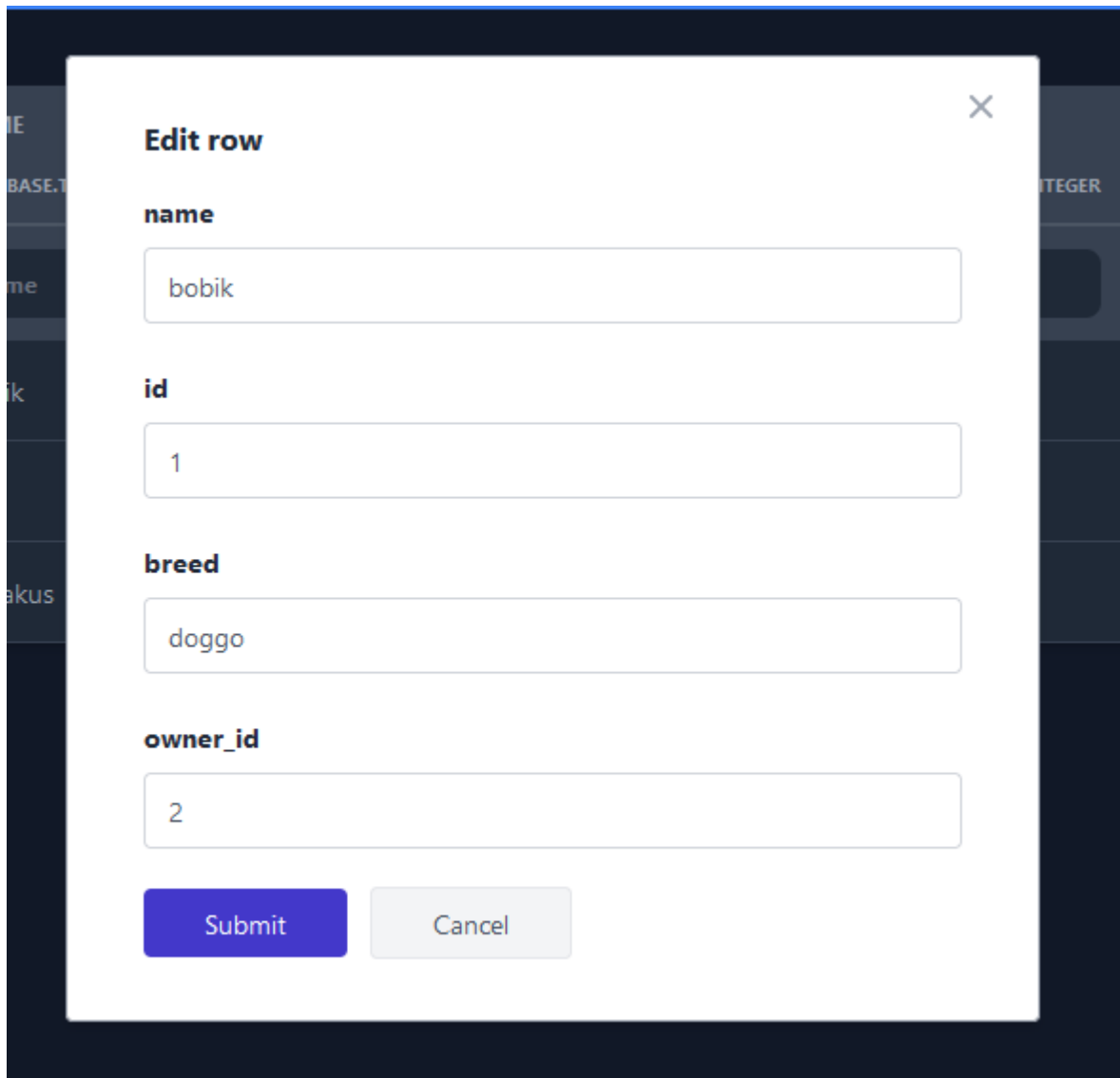


Таблиця бази даних Доступні операції

- Додати рядок
- Редагувати рядок
- Видалити рядок

Animals/Dogs					
NAME	ID	BREED	OWNER_ID		
DATABASE.TYPESTRING	DATABASE.TYPEINTEGER	DATABASE.TYPESTRING	DATABASE.TYPEINTEGER		
<input type="text" value="name"/>	<input type="text" value="id"/>	<input type="text" value="breed"/>	<input type="text" value="owner_id"/>	ADD	
bobik	1	doggo	2	Edit	Delete
juja	2	spaniel	3	Edit	Delete
sobakus	3	bulldog	1	Edit	Delete

Вікно редагування рядка



Edit row ✕

name

id

breed

owner_id

Submit **Cancel**

Вікно операції Join Tables

Select first table

Dogs

Select second table

owners

Select first column

owner_id :: database.TypeInteger

Select second column

id :: database.TypeInteger

Join tables

Результат операції Join Tables

Animals/JOINED						
NAME	ID	BREED	OWNER_ID	NAME	WEIGHT	LICENSE
database.TypeString	database.TypeInteger	database.TypeString	database.TypeInteger	database.TypeString	database.TypeReal	database.TypeChar
name	id	breed	owner_id	name	weight	license
bobik	1	doggo	2	maksym	69	82
jaja	2	spaniel	3	kira	47.7	81
sobakus	3	bulldog	1	oleg	123	72

3. Посилання на GitHub репозиторій

https://github.com/Rasakhatskiy/Labs_S7_IT