

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ  
ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ СИСТЕМ І ТЕХНОЛОГІЙ  
«КІЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ імені  
ВАДИМА ГЕТЬМАНА»  
ДЕРЖАВНОГО ВИЩОГО НАВЧАЛЬНОГО ЗАКЛАДУ

**Курсовий проект**  
з дисципліни  
проектування інформаційних систем  
**«Проектування IC для роботи за мовою DDLog «Play DDLog»**

Виконав: студент 4 курсу 403 групи  
Бакаев Артем Олександрович

Приняв:  
викладач, д.е.н.  
професор  
Мозгалли О.П.

Київ – 2022

## ЗМІСТ

<b>Аннотація</b> .....	3
<b>Вступ</b> .....	4
<b>Розділ 1 Характеристика та аналіз предметної галузі</b> .....	5
1.1 Характеристика предметної області та об'єкта дослідження .....	5
1.2 Дослідження існуючих інформаційних систем для обраної пре- дметної області .....	5
1.3 Розробка концепції інформаційної системи .....	7
<b>Розділ 2 Обґрунтування методології проектування та постановка за-     дачі</b> .....	8
2.1 Обґрунтування методології проектування та функціональна мо- дель задачі .....	8
2.2 Характеристика задачі.....	8
2.3 Вихідна інформація .....	8
2.4 Вхідна інформація .....	9
2.5 Математичне забезпечення та алгоритм функціонування системи .	10
<b>Розділ 3 Розробка рішень для ІС за видами забезпечення</b> .....	11
3.1 Інформаційне забезпечення.....	11
3.2 Організаційне забезпечення .....	12
3.3 Програмне забезпечення .....	13
3.4 Технічне забезпечення .....	13
<b>Висновки</b> .....	15
<b>Додатки</b> .....	17
3.5 Додаток А .....	17
3.6 Додаток Б .....	21
3.7 Додаток В .....	22

## АННОТАЦІЯ

Головною ідеєю курсового проекту було проектування ІС для роботи з мовою програмування DDLog.

Проектована система уможливить швидке прототипування логічних моделей та злегшить їх тестування.

В даному курсовому проекті було проведено аналіз предметної області, дослідження систем-конкурентів а також спроектовано ІС для роботи з мовою DDLog «Play DDLog» (далі просто IC).

Були приведені архітектурні діарами стандарту UML. Проведено дизайн БД для IC.

## ВСТУП

Тема ІС для інтерактивної роботи з логічними моделями є актуальною, оскільки з проведеного дослідження предметної області видно, що для саме цього використання ІС створено не було.

Метою курсового проекту є проектування ІС для подальшої її реалізації в рамці бакалаврського проекту.

Для досягнення данної мети було визначено наступні завдання.

1. Проектування модулів для виконання основного функціоналу;
2. Проектування модулів і БД для підтримки обробки користувачів за для надання звичного функціоналу систем розробки;
3. Прототипування графічного інтерфейсу для кращого розуміння яка існує вхідна та вихіда інформація за межами основної місії ІС.

Були приведені архітектурні діаграми стандарту UML. Проведено дизайн БД для ІС.

При розробці курсового проекту було використано:

1. ОС - Windows 10;
2. текстовий редактор - LaTeX;
3. середовище проектування - UML Designer 9.0;
4. десктоп версія додатку Draw.io.

Робота складається із вступу, трьох розділів, висновків, додатків та переліку джерел посилання. В першому розділі буде дано характеристику предметної області, проведено порівняння існуючих рішень і виявлення їх особливостей. В другому розділі буде обрано методологію проектування ІС, наведено скріншоти функціональної моделі задачі, описано вхідну і вихідну інформацію. В третьому розділі буде наведено загальну схему інформаційних зв'язків і перелік елементів, які будуть використані в інформаційній системі.

## РОЗДІЛ 1

### ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

**1.1 Характеристика предметної області та об'єкта дослідження**  
 Мова DDlog є сучасним діалектом мови Datalog.

Мова Datalog була створена для зручної роботи з логічними моделями. Проте ряд істотних недоліків залишив цю мову в стані академічного інструменту для дослідників. Цими недоліками були:

- Відсутність системи типів, що істотно підвищувало складність і створення і відладки програм
- В оригінальній версії мова Datalog не мала ніяких способів виконувати процедури, тому навіть прості на перший погляд логічні конструкції могли мати дуже нетривіальні реалізації.
- В більшості реалізацій моделі виконувалися інтерпретаторами, це призводило до зниження ефективності роботи, що в кінці робило неможливим використання тих систем навіть на масштабі локальних БД (SQLite).

Разом з тим, було розвинено реляційну модель та створено перші реляційні СКБД, цей прорив витіснив мову Datalog з індустрії на багато років. Реляційна модель є виразною через конструкції логічні конструкції мови Datalog, тому є підмножиною логічної мови. З часом були виявлені недоліки реляційних моделей що призвело до виникнення парадігми NewSQL, та звернули погляди фахівців і науковців назад до мови Datalog.

**1.2 Дослідження існуючих інформаційних систем для обраної предметної області**

Існує багато програмних засобів для роботи за мовою Datalog та її розширеннями. Datalog в його багато численних реалізаціях зазвичай не має системи типів, а якщо і має то зазвичай базову, без можливості декларації своїх типів даних.

Проте, я хотів би звернути увагу на декілька з них:

1. Flix[3] - мова програмування що реалізує можливості логічного програмування через вбудовану підмову - діалект Datalog. Одночасно плюсом та мінусом є те, що це повноцінна мова програмування, з цього випливає те, що

не існує легкого способу експортувати моделі як компоненти, проте завдяки цьому можливо швидке прототипування логічних моделей.

2. Souffle[5]. Основне призначення - створення проміжних програмних компонентів для використання в компільованих программах. Теж діалект Datalog. Плюсом є те, що результатом використання є оптимізований C++ код, який можна застосовувати як завгодно.
3. DDLog[2]. Основне призначення - створення проміжних інкрементальних програмних компонентів для використання в компільованих программах. Мова DDlog має можливість вираховувати зміни в вихідних фактах при внесенні змін в вхідні факти. Це дозволяє побудувати реактивну логічну модель з її декларативного опису. Ця система генерує модулі мови Rust, що передбачують роботу в розподіленному середовищі.

Табличне порівняння:

Таблиця 1.1

Порівняння існуючих систем

Назва	Сценарій використання (основний)	Інкрементальність
Flix	Мова програмування	Не має
Souffle	Програмні компоненти	Не має
DDLog	Програмні компоненти	Є

З приведеного вище порівняння, можна сказати що нині, чистих та сучасних Datalog систем орієнтованих на неопосередковану роботу з логічними моделями Datalog не існує.

### 1.3 Розробка концепції інформаційної системи

В запропонованій системі «Play DDlog» використовується діалект Datalog DDlog. В якості інтерфейсу користувача обрано веб інтерфейс тому що він гарантує максимальну доступність результируючого додатку.

Скетч інтерфейсу користувача наведено на рис. 1.1

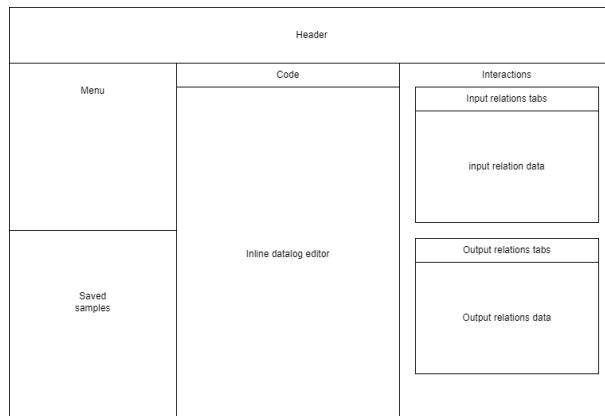


Рисунок 1.1 – Скетч інтерфейсу користувача

## РОЗДІЛ 2

### ОБГРУНТУВАННЯ МЕТОДОЛОГІЇ ПРОЕКТУВАННЯ ТА ПОСТАНОВКА ЗАДАЧІ

#### 2.1 Обґрунтування методології проектування та функціональна модель задачі

Методологією проектування обрана модель waterfall [4], тому що предметна область не відрізняється високою динамікою на ринку а тому і в плані вимог. Обрана модель забезпечує оптимальний процес створення системи.

Стандартом за яким проводиться опис ІС було обрано UML, середовище - UML Designer 9.0

Функціональна модель системи наведено ниże:

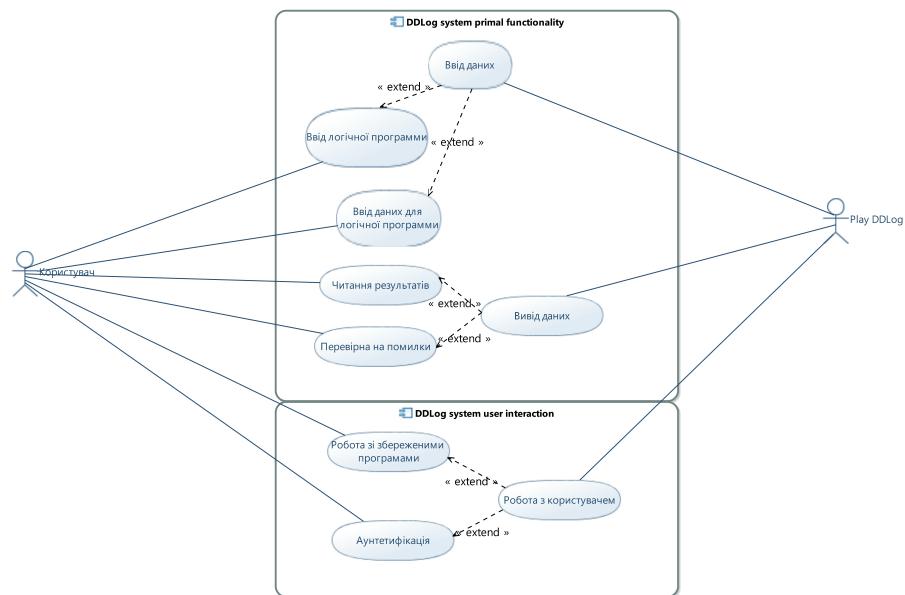


Рисунок 2.1 - UML Use Case

#### 2.2 Характеристика задачі

ІС призначена для зручної роботи з логічними моделями мови DDLog. Це дозволяє досить швидко прототипувати моделі на цій мові. Персонал що працює з системою може використовувати її в будь-який час для тестування та використання програм DDLog.

Для зручності, система також повинна підтримувати роботу декількох користувачів, кожного з можливістю зберігати поточні моделі в ІС.

Інформаційна модель приведена нижче:

#### 2.3 Вихідна інформація

Вихідна інформація ІС:

1. Інтерактивна логічна модель
2. Повідомлення про допущені в коді помилки, якщо ті були
3. Результати взаємодії з логічною моделлю

Деталі в таблиці відповідно номерам:

Таблиця 2.1

#### Вихідна інформація IC

№ з/п	Ідентифікатор	Форма подання	Допустимий час затримки	Користувачі інформації
1	model	Елемент веб сторінки	1-2 минути	Кінцевий користувач
2	errors	Елемент веб сторінки	1-2 минути	Кінцевий користувач
3	results	Елемент веб сторінки	1-2 минути	Кінцевий користувач

#### 2.4 Вхідна інформація

Вхідна інформація IC:

1. Код логічної моделі.
2. Будь-які вхідні дані цієї логічної моделі
3. Запит на збереження логічної моделі в IC БД
4. Запит на відтворення збереженої
5. Логін-пароль для ідентифікації користувача.

Деталі в таблиці відповідно номерам:

Таблиця 2.2

#### Вхідна інформація IC

№ з/п	Ідентифікатор	Форма подання	Джерело
1	source	Елемент веб сторінки	Кінцевий користувач
2	input	Елемент веб сторінки	Кінцевий користувач
3	save	Елемент веб сторінки	Кінцевий користувач
4	load	Діалог на веб сторінці	Кінцевий користувач
5	login	Веб сторінка	Відвідувач веб сторінки

## 2.5 Математичне забезпечення та алгоритм функціонування системи

Робота з мовою програмування DDLog передбачає обробку її коду. Це включає в себе:

- Токенізація
- Побудова абстрактного синтаксичного дерева
- Передобробка AST
- Побудова словника правил, створення БД фактів.

Перші три етапи можуть завершитися помилками, які будуть виведені користувачеві.

В результаті обробки коду, ми отримуємо опис логічної моделі. Цей опис надалі використовується ІС для інтерактивної роботи з нею.

Нижче наведено UML Sequence діаграмму, що ілюструє процес роботи системи

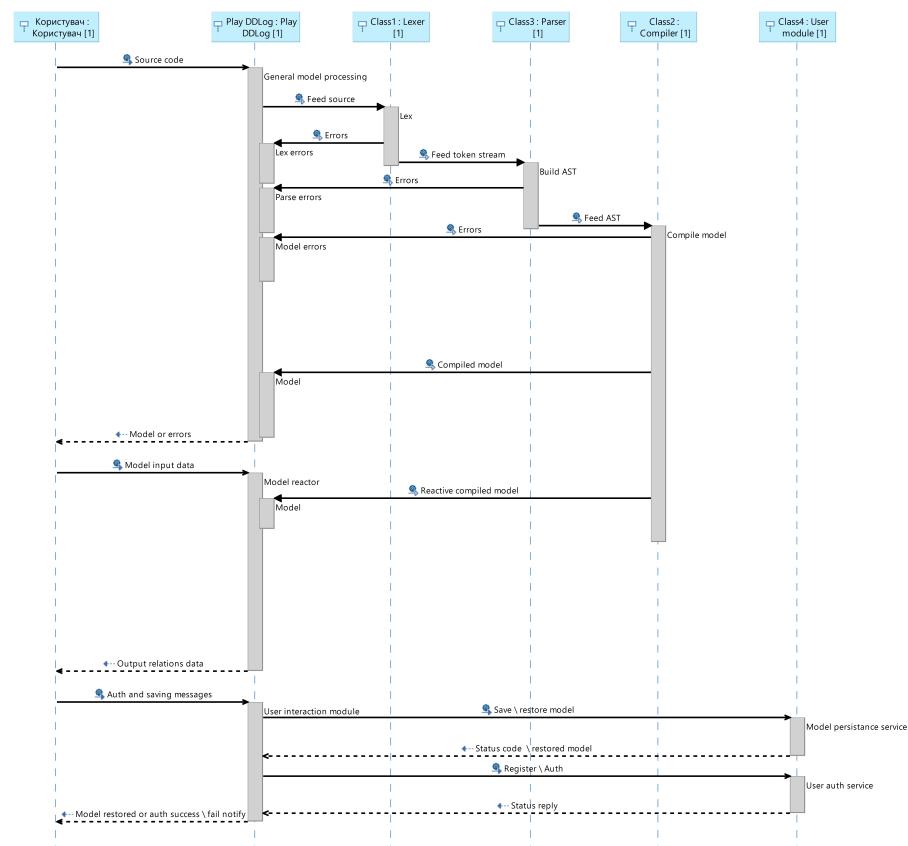


Рисунок 2.2 - UML Sequence

Зразок коду лексера надано в додатку А.

## РОЗДІЛ 3

### РОЗРОБКА РІШЕНЬ ДЛЯ ІС ЗА ВИДАМИ ЗАБЕЗПЕЧЕННЯ

#### 3.1 Інформаційне забезпечення

Для роботи ІС з користувачами - підтримки функціоналу зберігання і відтворення моделей системи використовує SQL БД.

Вибір БД є не принциповим - підійде будь-яка БД що підтримує SQL:2006.

Структура БД наведена нижче:

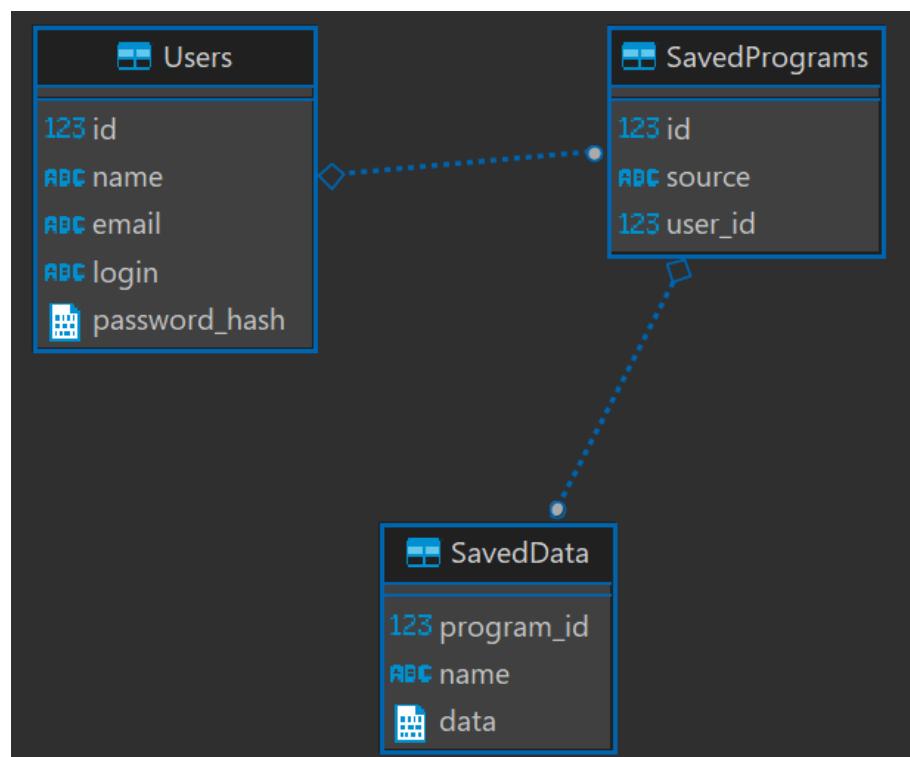


Рисунок 3.1 - Схема таблиць бази даних

Код SQL для створення БД наведено в додатку Б. Декілька релевантних запитів на мові SQL наведено в додатку В.

ІС для коректного функціонування та збереження можливості подальших змін повинна бути поділена на модулі.

Перелік, взаємозв'язки та призначення модулів наведено нижче:

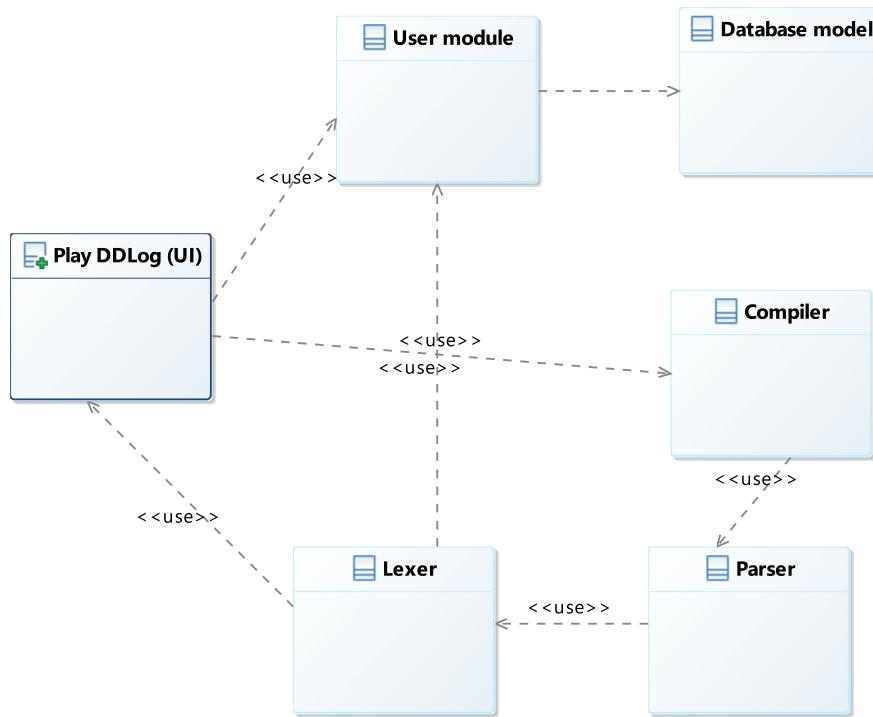


Рисунок 3.2 - Діаграма класів (для бізнес логіки)

Для взаємодію браузерної частини IC з серверною обрано архітектурний підхід REST а за форматом кодування обрано JSON.

Конкретні типи повідомлень майже неможливо розробити наперед, а тому вони повинні бути визначені на етапі розробки, тут їх визначено не буде.

### 3.2 Організаційне забезпечення

При взаємодії з системою можна виділити ключові точки, що зображені на діаграммі станів нижче.

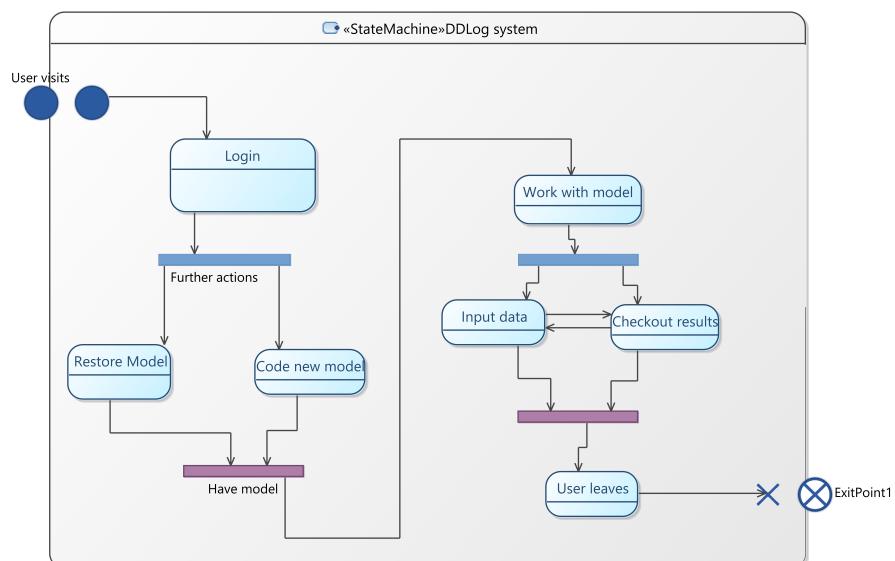


Рисунок 3.3 - Діаграмма станів користувача

## Користувач

### 3.3 Програмне забезпечення

При проектуванні програмного забезпечення мають бути вироблені рішення щодо системного і прикладного програмного забезпечення. Для ілюстрації структури прикладного програмного забезпечення наводяться діаграмами компонентів (UML).

Окрім внутрішньої організації модулів для змістової частини ІС існує ще невідємна потреба в інфраструктурних модулях ІС. Наприклад - робота з веб протоколами.

Загальну структуру компонентів ІС наведено нижче:

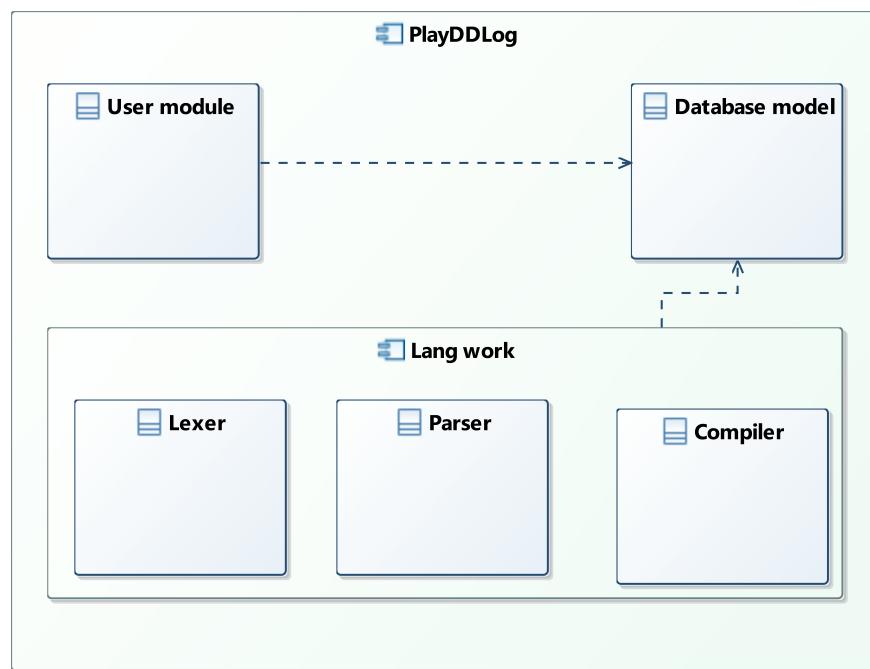


Рисунок 3.4 - Діаграмма компонентів

Конкретний код ІС та цього документу знаходиться в git-репозиторії за посиланням: [1]

### 3.4 Технічне забезпечення

Комплекс технічних засобів обрано як для розгортання веб додатку. Це включає в себе:

1. Сервер що обслуговує запити на веб сторінки;
2. Сервер що обслуговує запити до API;

### 3. СКБД для роботи з БД.

Кінцевий вигляд розгорнутої системи наведено нижче:

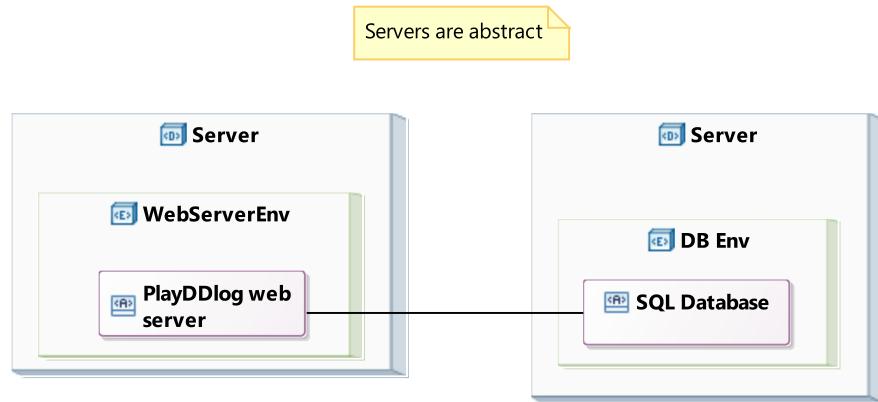


Рисунок 3.5 - Діаграмма розгортання

Взагалі, всі компоненти ІС можуть виконуватися як на одній, так і на різних комп’ютерах. Вони можуть бути як фізичні, так і віртуальні (ВМ). Весь комплекс ПЗ може виконуватись як в рамках однієї ВМ, так і на великому комплексі серверів.

## ВИСНОВКИ

В першому розділі було охарактеризовану предметну область систем для роботи з мовою Datalog взагалі, та її діалектом DDlog конкретно, досліджено існуючі інформаційні системи та розроблено концепцію інформаційної системи.

В другому розділі було обґрунтовано методології проектування та функціональних моделі задачі, складено характеристику задачі, описано вхідну і вихідну інформацію, описано математичне забезпечення та алгоритм функціонування системи.

В третьому розділі було описано інформаційне забезпечення, організаційне забезпечення, програмне забезпечення та технічне забезпечення і надано відповідні діаграми.

Розроблений проект є життєздатним і може бути використаним в майбутньому.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- [1] *GitHub repository*. 2021. URL: [https://github.com/tema3210/webddlog\\_study](https://github.com/tema3210/webddlog_study).
- [2] Oracle inc. та community. *GitHub repository*. 2021. URL: <https://github.com/vmware/differential-datalog>.
- [3] M. Madsen. *An Introduction to the Flix Programming Language*. 2022. URL: <https://flix.dev/talks/dahl-nygaard.pdf>.
- [4] *Опис моделі waterfall*. 2019. URL: <https://encyclopedia.thefreedictionary.com/waterfall+model>.
- [5] *Офіційний веб-сайт*. 2016. URL: <https://souffle-lang.github.io/docs.html>.

## ДОДАТКИ

### 3.5 Додаток А

```

#[inline]
pub(crate) fn lexer(inp: &str) -> Result<Vec<Token>,AppError>{
    let subber = |item: &str, mut last_ended_with_number: bool| -> Result<Vec<Token>,AppError> {
        let mut ret = Vec::new();
        let mut it = item.chars().peekable();

        let op_pred = |ch: char| -> bool {
            ['+', '-', '*', '/', '^'].iter().position(|&c| c == ch).is_some()
        };
        let brace_pred = |ch: char| -> bool {
            if ch == '(' || ch == ')' { true } else { false }
        };

        //number before dot, number after dot, position of dot?, presence of number?, is negative?
        let mut num_state: (f64,f64,Option<i32>,bool,bool) = (0.0,0.0,None,false,false);

        // let mut op_cache: Option<char> = None;

        let dump numb = |state: (f64,f64,Option<i32>,bool,bool)| -> f64 {
            let sgn: f64 = if state.4 { -1.0 } else { 1.0 };
            let ret = if let Some(shift) = state.2 {
                state.0 + state.1 * 10.0f64.powi(-shift)
            } else {
                state.0
            };
            ret * sgn
        };
        loop {

```

```

match it.next() {
    Some(ch) if op_pred(ch) => {
        //is in number? is number negative?
        match (num_state.3,num_state.4) {
            (true,_) => {
                ret.push(Token::Num(dump_numb(num_state)));
                num_state = (0.0,0.0,None,false,false);
                ret.push(Token::Op(ch));
            },
            (false,true) => {
                unreachable!("negative flag outside of number")
            },
            (false,false) if ch == '-' => {
                if last_ended_with_number {
                    ret.push(Token::Op(ch));
                    last_ended_with_number = false;
                } else {
                    match it.peek() {
                        Some(ch) if ch.is_digit(10) => {
                            num_state.4 = true;
                            num_state.3 = true;
                        },
                        _ => {
                            ret.push(Token::Op(ch));
                        },
                    };
                };
            },
            (false,false) => {
                ret.push(Token::Op(ch))
            }
        }
    },
}

```

```

Some(ch) if brace_pred(ch) => {
    if num_state.3 {
        ret.push(Token::Num(dump_numb(num_state)));
        num_state = (0.0,0.0,None,false,false);
    }
    ret.push(Token::Brace{lhs: ch == '('});
},
Some(ch) if ch.is_digit(10) => {
    num_state.3 = true;
    if let Some(ref mut dp) = num_state.2 {
        num_state.1 = num_state.1 * 10. + ch.to_digit(10).unwrap() as f64;
        *dp+=1;
    } else {
        num_state.0 = num_state.0 * 10. + ch.to_digit(10).unwrap() as f64;
    }
},
Some('.') => {
    if num_state.3 == false {break Err(AppError::LexError("Found dot outside of number"))}
    if num_state.2.is_some() {
        break Err(AppError::LexError("Found number with 2 dots".into()))
    } else {
        num_state.2 = Some(0)
    }
},
Some(_) => {
    break Err(AppError::LexError("Found improcessable char".into()))
},
None => {
    if num_state.3 {
        ret.push(Token::Num(dump_numb(num_state)))
    }
    break Ok();
}

```

```
        }
    }?;

Ok(ret)
};

inp.split(' ').filter(|s| !s.is_empty()).try_fold(Vec::with_capacity(inp.len()/2),|mut acc,it|{
    let flag = if let Some(item) = acc.last() {
        matches!(item,Token::Num(_))
    } else {
        false
    };
    acc.append(&mut subber(it,flag)?);
    Ok(acc)
}).map(|mut ok| {Vec::shrink_to_fit(&mut ok);ok})
}
```

### 3.6 Додаток Б

```
CREATE TABLE Users (
    id int(10) primary key auto increment,
    name varchar(50),
    email varchar(50),
    login varchar(50),
    password_hash blob
)
```

```
CREATE TABLE SavedPrograms (
    id int(10) primary key auto increment,
    user_id int(10)
    source blob,
    FOREIGN KEY (user_id) REFERENCES Users(id)
)
```

```
CREATE TABLE SavedData (
    program_id int(10),
    name varchar(50),
    data blob,
    CONSTRAINT SavedDataPK PRIMARY KEY (program_id,name),
    FOREIGN KEY (program_id) REFERENCES SavedPrograms(id)
)
```

### 3.7 Додаток В

Запит на хеш пароля користувача:

```
SELECT password_hash FROM Users WHERE login = \%\%login\%\%;
```

Запит на id збережених програм користувача:

```
SELECT id FROM SavedPrograms WHERE user_id = \%\%id\%\%;
```

Запит на збережену програму:

```
SELECT * FROM SavedPrograms WHERE id = \%\%id\%\%;
```

Запит на збережені програми та їх дані:

```
SELECT
    SavedPrograms.id as 'id',
    SavedPrograms.source as 'source',
    JSON_ARRAYAGG(JSON_OBJECT('name', SavedData.name, 'data', SavedData.data))
FROM SavedPrograms JOIN SavedData ON SavedData.program_id = SavedPrograms.id
WHERE SavedPrograms.user_id = \%\%user_id\%\%
GROUP BY SavedProgram.id;
```