**Теоретична частина**

Тестирование теоретических знаний

1. Что выведется на экран?

class ClassA {}

$a = new ClassA();

$b = new ClassA();

var\_dump($a === $b);

Варианты ответов:

**bool(false)**

bool(true)

Вылетит ошибка.

2. Что выведется на экран?

var\_dump([

"a" => 1,

"b" => 2,

"c" => [1,2,3]

] === [

"a" => 1,

"b" => 2,

"c" => [1,2,3]

]);

Варианты ответов:

bool(false)

**bool(true)**

Вылетит ошибка.

3. Что выведется на экран?

class ClassA {

public int $var1;

public function \_\_construct(int $var1)

{

$this->var1 = $var1;

}

}

$a = new ClassA(5);

$b = $a;

$b->var1 = 10;

echo $a->var1;

Варианты ответов:

5

**10**

Вылетит ошибка.

4. Что выведется на экран?

class ClassA {

public int $var1;

public function \_\_construct(int $var1)

{

$this->var1 = $var1;

}

}

$a = new ClassA(5);

$b = $a;

$a->var1 = 10;

$a = null;

echo $b->var1;

Варианты ответов:

5

**10**

Вылетит ошибка.

5. Что выведется на экран?

class ClassA {

public int $var1;

public function \_\_construct(int $var1)

{

$this->var1 = $var1;

}

}

$a = new ClassA(5);

$b = &$a;

$b->var1 = 10;

$a = null;

echo $b->var1;

Варианты ответов:

5

10

**Вылетит ошибка.**

6. Что выведется на экран?

$numbers = [1,2,3,4,5];

foreach ($numbers as $number) {

$number \*\*= 2;

}

echo $numbers[4];

Варианты ответов:

**5**

10

25

Вылетит ошибка.

7. Что выведется на экран?

$numbers = [1,2,3,4,5];

foreach ($numbers as &$number) {

$number \*\*= 2;

}

$number = 10;

echo $numbers[4];

Варианты ответов:

5

**10**

25

Вылетит ошибка.

8. Что выведется на экран?

interface Movable {

public function move(): void;

}

interface Breathable {

public function breath(): void;

}

class Animal implements Movable, Breathable {

public function move(): void

{

echo self::class . ' move';

}

}

class Dog extends Animal {

public function move(): void

{

echo self::class . ' move';

}

}

(new Dog())->move();

echo "\n";

(new Animal())->move();

Варианты ответов:

Dog move

Animal move

Dog move

Dog move

Animal move

Animal move

**Вылетит ошибка.**

9. Каков результат выполнения данного кода?

interface Movable {

public function move(): void;

}

interface Breathable {

public function breath(): void;

}

interface Dog extends Movable, Breathable {

public function bark(): void;

}

Варианты ответов:

**Выполнится успешно.**

Вылетит ошибка.

10. Что выведется на экран?

interface Movable {

public function move(): void;

}

interface Breathable {

public function breath(): void;

}

class Animal implements Movable, Breathable {

public function move(): void

{

echo static::class . ' move';

}

public function breath(): void

{

echo static::class . ' breath';

}

}

class Dog extends Animal {

}

$dog = new Dog();

$dog->move();

echo "\n";

$animal = new Animal();

$animal->move();

Варианты ответов:

**Dog move**

**Animal move**

Dog move

Dog move

Animal move

Animal move

Вылетит ошибка.

11. Что выведется на экран?

interface Movable {

public function move(): void;

}

interface Breathable {

public function breath(): void;

}

class Animal implements Movable, Breathable {

public function move(): void

{

echo self::class . ' move';

}

public function breath(): void

{

echo self::class . ' breath';

}

}

class Dog extends Animal {

}

$dog = new Dog();

$dog->breath();

echo "\n";

$animal = new Animal();

$animal->breath();

Варианты ответов:

Dog breath

Animal breath

Dog breath

Dog breath

**Animal breath**

**Animal breath**

Вылетит ошибка.

12. Что сделает composer install команда?

Установит все зависимости согласно composer.json в vendor директорию.

Установит все зависимости согласно composer.json (даже если есть composer.lock файл) в vendor директорию,

после обновит composer.lock.

**Установит все зависимости согласно composer.lock (если есть) в vendor директорию, иначе использует composer. json и после создаст composer.lock файл.**

Установит все зависимости согласно composer.lock (если есть) в vendor директорию, иначе использует composer.

json.

13. Что сделает composer update команда?

Установит все зависимости согласно composer.json в vendor директорию.

**Установит все зависимости согласно composer.json (даже если есть composer.lock файл) в vendor директорию,**

**после обновит composer.lock.**

Установит все зависимости согласно composer.lock (если есть) в vendor директорию, иначе использует composer.

json и после создаст composer.lock файл.

Установит все зависимости согласно composer.lock (если есть) в vendor директорию, иначе использует composer.

json.

14. Какая composer команда установит новый пакет и добавит его в composer.json?

**composer require <имя пакета>**

composer install <имя пакета>

composer updatel <имя пакета>

composer add-dependency <имя пакета>

15. Какой сегмент (Обозначено как X) в версии композер пакета отвечает за версии без добавления нового

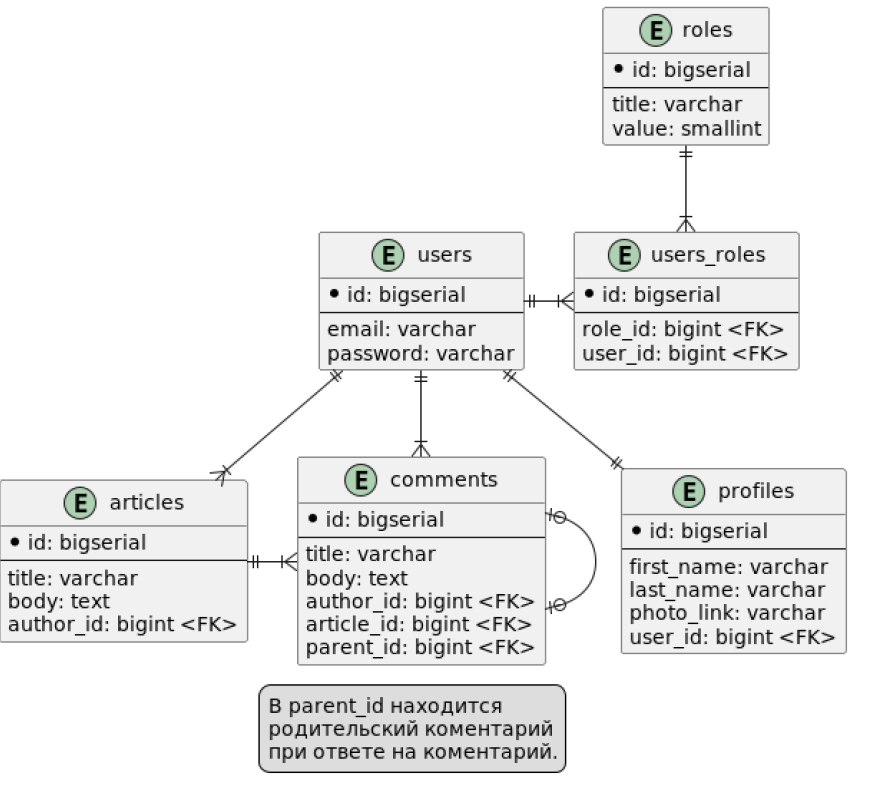
функционала и с сохранением обратной совместимости (без изменений интерфейса)?

X.2.3

1.X.3

**1.2.X**

Используя имеющуюся выше диаграмму напишите такие sql запросы



16. Достаньте все комментарии для статьи с id = 101, нужно достать такие столбцы:

article\_id

article\_title

article\_body

comment\_id

comment\_title

comment\_body

comment\_parent\_id

**SELECT (article\_id**, **article\_title, article\_body, comment\_id, comment\_title, comment\_body, comment\_parent\_id) FROM articles JOIN comments ON articles.id = comments.article\_id WHERE article.id = 101;**

17. Достаньте данные пользователя и все роли для пользователя с id = 256, нужно достать такие столбцы:

user\_id

email

first\_name

last\_name

photo\_link

role\_title

**SELECT (user\_id, email, first\_name, last\_name, photo\_link, role\_title) FROM users JOIN users\_roles JOIN roles JOIN profiles ON users.id = users\_roles.user\_id AND users\_roles.roles\_id = roles.id AND users.id = profiles.user\_id WHERE users.id** **= 256;**

18. Добавьте пользователю с id 225 новую роль (id роли = 8).

**INSERT INTO users\_roles VALUES(8,225);**

19. Обновите photo\_link для профиля пользователя с id = 67.

**UPDATE profiles SET photo\_link = “” WHERE user\_id = 67;**

20. Удалите пользователя с id = 78 со всеми его связями.

**DELETE FROM users WHERE user.id = 78;**

**(query works if “ON DELETE CASCADE” is stated in “users” table code)**

@startuml

entity "favourite" as f{

\* user\_id number <FK> <PK>

\* book\_id: number <FK> <PK>

}

entity "book\_author" as ba{

\* book\_id: number <FK> <PK>

\* author\_id: number <FK> <PK>

}

entity "book\_genre" as bg{

\* book\_id: number <FK> <PK>

\* genre\_id: number <FK> <PK>

}

entity "user" as u{

\* user\_email: varchar

--

user\_password: varchar

user\_position

}

entity "book" as b {

\* book\_id: serial

--

\*book\_name: text

book\_description: varchar

}

entity "author" as a{

\* author\_id: serial

--

\*author\_name: text

}

entity "genre" as g{

genre\_id: serial

--

\*genre\_name: text

}

b ||--|{ bg

g ||--|{ bg

b ||--|{ ba

a ||--|{ ba

u ||--o{ f

b ||--|{ f

@enduml

**Практична частина**

Тема: Web API онлайн-бібліотеки

Стек для розробки: ASP.NET Core

Вимоги до розробки:

Авторизация:

Регистрация с почтой и паролем.

Логин с почтой и паролем.

Книги:

список всех доступных - не требует логин;

получение информации по отдельной книге - не требует логина;

добавление новой книги - требует логин + роль админа;

удаление книги - требует логин + роль админа;

добавление книги в избранное конкретным пользователем (у каждого пользователя свой список избранных книг)

- требует логин;

удаление книги из избранного конкретным пользователем (у каждого пользователя свой список избранных книг)

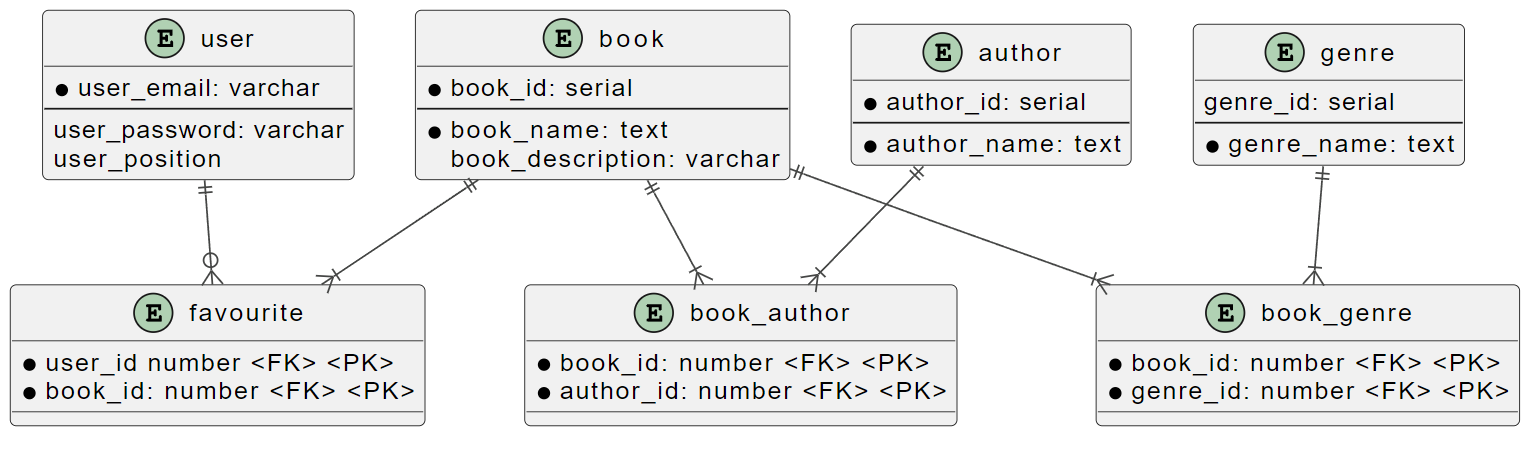
- требует логин + каждый пользователь имеет доступ только к своему списку;

выгрузка списка книг в csv формате - требует роль админа.

Роли:

возможность изменять роль для пользователя с помощью консольной команды;

доступные роли: администратор и клиент.



create TABLE users(

user\_email varchar(255) NOT NULL,

user\_password varchar(255) NOT NULL,

user\_position varchar(255) NOT NULL,

primary key(user\_email)

);

create TABLE books(

book\_id serial primary key NOT NULL,

book\_name varchar(255) NOT NULL,

book\_desc varchar(255)

);

create TABLE authors(

author\_id serial primary key NOT NULL,

author\_name varchar(255)

);

create TABLE genres(

genre\_id serial primary key NOT NULL,

genre\_name varchar(255)

);

create TABLE books\_genres(

book\_id int NOT NULL,

genre\_id int NOT NULL,

FOREIGN KEY (book\_id) REFERENCES books(book\_id),

FOREIGN KEY (genre\_id) REFERENCES genres(genre\_id),

primary key(book\_id, genre\_id)

);

create TABLE books\_authors(

book\_id int NOT NULL,

author\_id int NOT NULL,

FOREIGN KEY (book\_id) REFERENCES books(book\_id),

FOREIGN KEY (author\_id) REFERENCES authors(author\_id),

primary key(book\_id, author\_id)

);

create TABLE favs(

user\_email varchar(255) NOT NULL,

book\_id int NOT NULL,

FOREIGN KEY (user\_email) REFERENCES users(user\_email),

FOREIGN KEY (book\_id) REFERENCES books(book\_id),

primary key(user\_email, book\_id)

);