1. Як працюють індекси у БД?

   Індекси БД – це структура даних, яка призначена для пришвидшення пошуку в таблиці, з колонками якої індекси пов’язані.

   Для того щоб почати роботу з індексами, на прикладі MySQL, їх необхідно створити для певної колонки або групи колонок.

   Створена індексована структура даних займає додаткове місце у файловому сховищі, але в той же час зменшує кількість операцій записів та зчитувань, що стосується взаємодії з індексованою таблицею.

   Таблиця індексів оптимізована для виконання бінарного пошуку, що дозволяє кардинально зменшити час для отримання необхідної інформації. Однак така структура потребує підтримки, тобто постійного оновлення. Кожна операція додавання, оновлення чи видалення інформації у основній таблиці спричиняє оновлення у таблиці індексів, що у свою чергу може вплинути на загальну швидкодію системи.

2. Які є JOIN в MySQL?

   В MySQL існують наступні види JOIN: INNER JOIN (JOIN), LEFT JOIN, RIGHT JOIN, SELF JOIN, CROSS JOIN

   - **INNER JOIN (JOIN)** – повертає лише спільні рядки лівої та правої таблиці.
   - **LEFT JOIN** – повертає усі рядки лівої таблиці і рядки правої таблиці, які відповідають вказаній умові об’єднання (можливі NULL значення).
   - **RIGH JOIN** – повертає усі рядки правої таблиці і рядки лівої таблиці, які відповідають вказаній умові об’єднання (можливі NULL значення).
   - **SELF JOIN** – використовується для об’єднання таблиці з нею ж (під час об’єднання необхідно вказувати псевдоніми, а не дублювати назву таблиці).
   - **CROSS JOIN** – об’єднання всіх рядків лівої таблиці з усіма рядками правої таблиці.

3. Що таке dependency inversion?

   Dependency inversion – один із принципів парадигми SOLID, який говорить про те, що модулі вищого рівня не повинні залежати від конкретних модулів нижчого рівня. Натомість вони повинні залежати від абстрактного представлення певного функціоналу. Наприклад, від інтерфейсу та від класу, який його наслідує.

4. У чому різниця між GET та POST запитами?

   Різниця між GET та POST запитами полягає у:

- **Головному призначенні** – GET запит призначений для отримання даних. В той час, як головне призначення POST запиту – передача даних з подальшим збереженням, або оновленням.
- **Способі передачі даних –** GET запит передає дані в якості параметрів, представлених в URL адресі. POST запит передає інформацію у більш безпечний спосіб – у тілі запиту.
- **Кешуванні –** GET запит кешується у браузері клієнта та на проксі сервері. Також відповідь на цей запит, за деяких умов, кешується на сервері – для пришвидшення повторної відповіді. У свою чергу POST запит не кешується на стороні клієнта, але, за певних умов, відповідь на цей запит може кешуватись на сервері.

5. У чому різниця self та static?

   Якщо розглядати ***self*** та ***static*** у контексті функціоналу та призначення ***self***, різниця між ними буде полягати у ступені зв'язування. Коли ***self*** – інструмент раннього статистичного зв'язування, а ***static*** – пізнього. Це пов'язано з тим, що ***self*** визначається на етапі парсингу скрипту, а ***static*** – на етапі компіляції. Тому ***self*** завжди буде буквально відтворювати назву класу, у якому був оголошений. У той час ***static*** завжди набуває контексту того класу, у якому був викликаний. Навіть за умови, якщо він був оголошений у класі предку, а викликаний у класі нащадку.
   Також функціонал ***static*** не обмежується класовим зв'язуванням. За допомогою цього службового слова можна оголошувати статистичні змінні в контексті зони видимості конкретного класу, або функції, які зберігатимуть свої значення при повторному виклику контексту.
   Static також дозволяє створювати властивості та методи класів, до яких можна звертатися без створення екземпляру класу.

6. Яка різниця між MVP та MVC?

   Основна різниця між цими двома архітектурними шаблонами полягає у розподіленні обов'язків між їх ключовими ланками. Архітектура MVC (Model View Controller) розподіляє бізнес-логіку між моделлю та контролером майже порівну і дозволяє представленню (view) обмінюватись даними напряму з моделлю (прикладом є реалізація можливості використання екземпляру класу моделі в роутері і передачі інформації з бази даних прямо у представлення). Натомість, архітектура MVP (Model View Presenter) забезпечує взаємодію представлення лише з презентером. Останній, у свою чергу, концентрує в собі майже всю бізнес-логіку і перетягує на себе задачу по підготовці даних для представлення.

7. Яка різниця між == та ===?

   Відмінність між == та === полягає у тому, що подвійне дорівнює – це перевірка на рівність з урахування приведення типів, а потрійне дорівнює – це перевірка на рівність без приведення типів, тобто перевірка на ідентичність.

8. Що таке final class та final method?

   Клас оголошений з використанням ***final*** неможливо наслідувати. Метод оголошений з використанням ***final*** не може бути перевизначений.

9. З якими фреймворками працювали?

   Маю невеликий досвід роботи з Yii2. На даний момент працюю з Laravel. Приклади робіт на Laravel можна побачити за посиланнями нижче:

   https://github.com/khomyart/magenta-inventory-system

   https://github.com/khomyart/datasets_lmr_optimised

   https://github.com/khomyart/mayor-report-content-manager

   https://github.com/khomyart/mayor-report

10. Що можете розповісти про свої навички, що дають вам перевагу? Які з них відповідають нашим вимогам

    Робота, на якій я зараз знаходжусь (головний спеціаліст відділу інформаційних технологій), часто потребує від мене різних навичок – від вміння налаштування серверів до написання скриптів, парсерів, додатків різної складності. Потрібно постійно вчитися і вчити інших людей взаємодіяти із системами, які сприяють оптимізації їхнього робочого процесу.

    З технічних навичок, які відповідають вашим вимогам, я можу відмітити:

    - володіння PHP (чистий, або Laravel), JS (чистий, або Vue.js), CSS, HTML5, GIT
    - вміння працювати з API, а також створювати API
    - вміння аналізувати дані з API та інтегрувати ці дані у систему, з якою працюю
    - навички написання парсерів
    - взаємодії з unix системами
    - володіння MySQL
    - розуміння протоколів взаємодії HTTP, HTTPS, RESTful API

    А стосовно особистих навичок я можу відмітити наполегливість, бажання вчитися, розвиватися у тій сфері, яку обрав. Люблю ділитися знаннями з іншими і допомагати у вирішенні проблем, які стосуються моєї спеціальності та прямих обов'язків. Відповідально відношусь до поставлених мені задач і стараюсь виконати їх якомога якісніше у властивій мені скрупульозній манері.