OGÓLNE

1. Czy pracowałeś w metodykach zwinnych? Jeżeli tak to w jakich?
Tak, w Scrum.

2. Czy wiesz czym jest Kanban? Czy pracowałeś w nim?
Tak, Kanban był stosowany przez PMów w zespołach,w których pracowałem w przerwach między etapami rozwojowymi projektu. Podczas realizacji części utrzymaniowych lub poaudytowych.

3. Jak oceniasz swój poziom języka angielskiego w mowie i piśmie?
Poziom komunikatywny. Pracuję w większości w języku angielskim i nie sprawiał on nigdy problemów w komunikacji (choć w mowie korzystam z niego bardzo rzadko).

SCM

1. Czy wiesz czym jest system kontroli wersji? Jeżeli tak to wyjaśnij w kilku słowach.
To system służący do kontroli zmian w kodzie dokonywanych członków zespołu. Umożliwia śledzenie zmian i ich historii i weryfikację autorów zmian. Ponadto ułatwia równoległą pracę przez możliwość „odbicia się” od tej samej wersji kodu i późniejsze scalanie do wspólnego brancha (co czasem wiąże się z koniecznością rozwiązania konfliktów).

2. Jeżeli korzystasz z systemu kontroli wersji to w jakiej formie i dlaczego?
Korzystałem z GITa z wykorzystaniem zewnętrznego repozytorium na Bitbuckecie i pipeline'ami na GitLabie.

3. W jaki sposób wyciągnąć pojedynczy commit? (GIT)
Jeżeli dobrze rozumiem pytanie, chodzi o przeniesienie commita między branchami. Służy do tego tzw. cherry pick:

```
git cherry-pick  #commit-hash
```

BAZY DANYCH

1. Z jakich baz danych korzystałeś?
MySQL/MariaDb
Czasem MySQL + Elasticsearch

2. Czy wiesz czym się różni baza relacyjna od obiektowej?
W ogólnym założeniu. Relacyjne bazy danych przechowują dane tabelach, które można łączyć.
Obiektowe bazy danych przechowują dane jako atrybuty obiektów. Nie korzystałem nigdy z obiektowych baz danych.

3. Czy potrafisz zaprojektować za pomocą diagramu bazę danych?
Kiedyś uczyłem się korzystania z narzędzi do tworzenia diagramów. W praktyce jednak nie wykorzystuję tego w codziennej pracy. Diagramy zdarza mi się czasem czytać, lecz raczej ich nie tworzę.

4. Czy wiesz czym są indeksy? Jakie są zalety i wady korzystania z indeksów?
Indeksy pozwalają na porządkowanie tabeli według określonej indeksem kolumny. Różne silniki baz danych pozwalają na korzystanie z różnych typów indeksów, jednak w praktyce wykorzystuję 2 z nich:

- Btree – sortuje tabelę według indeksowanej kolumny. Pozwala to na dostęp do wiersza analogicznie jak w b-drzewie, więc w czasie logarytmicznym. Pozwala na porównania =, >, >=, <, <=. Ponadto, znacząco przyspiesza operację ORDER BY.
- Hash – traktuje tabelę jak tablicę haszującą z indeksowaną kolumną jako zbiorem kluczy. Działa jedynie z operatorami =, <=>, ale czas dostępu do wiersza jest stały. Nie przyspiesza ORDER BY.

Stosowanie indeksów jest bardzo kosztowne w momencie zapisu do tabeli, więc należy nakładać dodatkowe indeksy tylko na tabele często odczytywane i rzadko zapisywane oraz na faktycznie wykorzystywane kolumny. Warto dodać, że poza komendą INDEX, indeks są także domyślnie nakładane podczas stosowania: PRIMARY KEY, UNIQUE, i FULLTEXT.

5. Czy dokonywałeś kiedyś jakiejś optymalizacji na bazie danych? Jeżeli tak to opisz na czym poległa.
Jeżeli chodzi o optymalizację zapytań było to min.:

- łączenie wielu zależnych zapytań w jedno duże z wykorzystaniem relacji (oczywiście należy wówczas kontrolować zużycie pamięci i zwykle pobierać dane w paczkach),
- wrzucenie kilku mniejszych zapytań do pojedynczej transakcji w celu zmniejszenia liczby połączeń z bazą,
- dodawanie indeksów,

6. Czy wiesz czym są transakcje? Po co je stosujemy?
Transakcja jest to wykonanie określonej liczby operacji w trakcie jednego połączenia z bazą danych. Istotą transakcji jest to, że operacje nie mogą zostać wykonane oddzielnie, ale wszystkie muszą zakończyć się powodzeniem.
Transakcje stosuję głównie z 2 powodów:

- zachowanie integralności i spójności danych – w ramach jednej transakcji wykonuje się operacje, które nie mają sensu oddzielnie lub wręcz wykonane niezależnie psułyby bazę,
- poprawa wydajności.

Transakcje powinny przestrzegać 4 podstawowych reguł, tzw. ACID: Niepodzielność, Spójność, Izolacja, Trwałość.

PROGRAMOWANIE - OGÓLNE PYTANIA

1. Czy przywiązujesz uwagę do formatowania kodu? Jeżeli tak to dlaczego?
Znany bon-mot mówi, że kod jest pisany tylko raz, a czytany wielokrotnie. I to jest główna przyczyna dla której dba się o formatowanie. Czytanie cudzego kodu (lub swojego napisanego dawno) jest łatwiejsze gdy jest on pisany według określonych reguł.
W PHP korzystam ze standardów PSR, wspomagając się czasem takimi narzędziami jak

PHPStan czy PHPCodeSniffer. Ponad to, PHPStorm znacznie ułatwia bieżące formatowanie.

2. Czy testujesz swój kod? Czy wiesz na czym polega TDD? Czy stosowałeś kiedyś takie podejście?
Testuję wówczas gdy budżet projektu i wycena godzinowa konkretnego zadnia to przewiduje (będąc szczerym – nie zdarza się to bardzo często). TDD polega na rozpoczęciu pracy od napisania testów, przed właściwym kodem. Znam założenia, ale nie miałem możliwości pracy w tej metodyce.

3. Czy trzymasz się jakichś standardów kodowania, jeśli tak to jakich?
PSR w PHP,
PEP w Pythonie.

4. Czy wiesz czym jest KISS ? Opisz i podaj negatywny i pozytywny przykład zastosowania.
Jest to zasada polegająca na pisaniu kodu w możliwie prosty (choć nie zbyt prosty) sposób.
Przykładem źle zastosowanej zasady KISS może być pomijanie wzorców projektowych czy rezygnacja ze stosowania interfejsów w celu pozornego uproszczenia kodu poprzez okrojenie go z warstw abstrakcji.
Pozytywnym przykładem użycia może być dzielenie złożonej logiki na mniejsze, odseparowane fragmenty i przenoszenie ich od osobnych funkcji/metod/klas.

5. Czy wiesz czym jest SOLID? Co według Ciebie jest w nim istotne?
Jest to zbiór 5 reguł programowania obiektowego. Najważniejsza wydaje się pierwsza z nich *Single-Responsibility Principle*, gdyż ma ona wpływ bezpośrednio na logikę i jej implementację z projekcie.

6. Jakie ryzyko może nieść używanie liczb zmienno-przecinkowych podczas programowania (float, double, etc)?
Pomijając czysto teoretyczne zagrożenia, takie jak niewystarczający zakres, które w webdevie zwykle nie występują, jako podstawowy problem wymieniłbym porównania, które mogą dawać mylące wyniki niezależnie od rozmiaru mantysy.
Np. przedstawione niżej wywołanie funkcji:

```
function compare(float $x, float $y): bool
{
     if ($x === $y) {
           return true;
     } else {
           return false;
     }
}

compare(1.0 - 0.8, 0.2);
```

może zwrócić (i zwykle zwróci) *False*, mimo iż prosta arytmetyka sugerowałaby, że wynikiem będzie *True*.

7a. Modyfikatory dostępu. Jak uważasz w jakich sytuacjach powinno się używać poszczególnych?
Może być to kwestia zależna od konkretnego języka i standardu,a nawet frameworka (np. dokumentacja Magento ściśle wskazuje kiedy, jakich modyfikatorów używać). Ogólny standard PSR-2 dla PHP określa, że widoczność musi być jawnie określona przy każdym polu oraz metodzie. Ponieważ ostatnio pracuję w większości w Magento stosuję przyjętą tam metodykę silnej hermetyzacji. Wszystkie atrybuty, metody oraz stałe powinny być prywatne. *Powinny* oznacza, że mają być prywatne, chyba że istnieje jasny powód aby nie były.

7b. Czym jest klasa abstrakcyjna? Do czego byś jej użył?
Klasa abstrakcyjna jest klasą, która nie tworzy obiektów a jedynie dostarcza logikę i interfejs dla klas pochodnych. Metody w klasie abstrakcyjnej mogą zawierać zarówno definicje jak i samą deklarację (metody abstrakcyjne). Klasa abstrakcyjna może także zawierać atrybuty, dziedziczyć po innych klasach abstrakcyjnych oraz implementować interfejsy.
Klas abstrakcyjnych można skutecznie używać kiedy chcemy dostarczyć metody klasom pochodnym a jednocześnie wymusić implementację innych metod poprzez narzucenie im interfejsu. Np. wzorce *Template Method* czy *Factory method*.

7c. Czym jest enkapsulacja? Czy uważasz że jest ważna i dlaczego?
Enkapsulacja, nazywana też hermetyzacją jest mechanizmem ograniczenia dostępu do części kodu poprzez nadanie odpowiednich modyfikatorów dostępu i ukrycie go pod abstrakcją. Popularnym wykorzystaniem mogą być gettery/settery w modelach czy hermetyzacja konstruktora w singletonie. Hermetyzacja bywa ważnym elementem poprawiającym bezpieczeństwo. Np. settery pozwalają nam uzyskać pewność, że odpowiednio zwalidujemy dane przed zapisem.

PHP

1. Czy wiesz czym jest Composer? Czy rozumiesz czym różni się update od install?
Composer służy do instalowania zależności w projekcie. Daje także możliwość wydawania własnych modułów, które chcemy wykorzystać w wielu projektach poprzez Packagist. Komenda *install* zainstaluje dokładnie te wersje pakietów jakie zapisane są w *composer.lock,* natomiast *update* podbije wersje pakietów do najnowszych na jakie pozwala zapis w *composer.json*.

2. Czy wersjonowałeś kiedyś schemat bazy danych? Jeśli tak to w jaki sposób?
Tak, w Magento 2 przy użyciu plików *db_schema.xml,* w Laravelu z wykorzystaniem migracji.

3. Jak byś zasilił bazę danych przykładowymi danymi?
To zależy od dostępnych narzędzi. Laravel dysponuje dedykowanymi narzędziami do wprowadzania przykładowych danych (połączenie *Seeders* z *Factories*). W Magento można napisać komendę CLI wprowadzającą dane (przez modele lub kwerendą raw SQL). Uniwersalnie można wygenerować dane w skrypcie php i połączyć się z bazą za pomocą biblioteki PDO.

4. Czy korzystałeś z Laravela, Symfony lub innego frameworka MVC? Jeżeli tak to jak długo i jak oceniasz swoją znajomość z w/w.
Przez ostatnie 2 lata pracowałem głównie we frameworku Magento 2, który też miejscami korzysta ze wzorca MVC.

Jeśli chodzi o typowy frameworki MVC, to posiadam niekomercyjne doświadczenie w Laravelu (piszę w nim okazjonalnie od 3 lat).

5. Jakie widziałbyś zastosowanie dla traitów?
Sporym ograniczeniem PHP jest fakt, że język ten nie pozwala na dziedziczenie po wielu klasach. Traity mogą pozwolić na obejście tego ograniczenia. Traity pozwalają na dodanie metod/atrybutów/stałych bez naruszania struktury hierarchii klas. Może być szczególnie użyteczne do domieszkiwania kodu do klasy finalnej, której nie możemy już rozszerzyć.

SYSTEMY OPERACYJNE (*NIX)

1. Czy korzystasz lub korzystałeś z jakiegoś systemu z rodziny *NIX? Jeżeli tak to z jakiego?
Pracuję na co dzień na Linuxie. Posługuję się nim biegle na poziomie użytkownika.

2. W jaki sposób sprawdziłbyś czy proces o danej nazwie jest nadal uruchomiony?
Komendą:

```
top
```

(trzeba przejrzeć listę) lub połączeniem *top* i *grep*:

```
top grep|nazwa
```

3. Czy wiesz czym się różni pamięć fizyczna od wirtualnej? Opisz.
Pamięć wirtualna to pamięć fizyczna (RAM) powiększona o partycję wymiany na dysku.

4. Czy wiesz czym jest swap i jaką pełni rolę?
W nawiązaniu do poprzedniego punktu, jest to wydzielona partycja na dysku (przestrzeń wymiany) służąca jako pamięć operacyjna w momencie wyczerpywania się miejsca w pamięci fizycznej.

5. Czym jest "load"?
Ogólnie, wskazuje średnie obciążenie systemu w kilku okresach czasu.
Nie wykorzystywałem tego nigdy jako narzędzia administracyjnego.

6. Czy wiesz w jaki sposób zrealizowane są prawa dostępu do plików i katalogów w systemach z rodziny *NIX? Opisz skrótowo.
Na poziomie użytkownika. Każdy z plików/katalogów posiada 3 kręgi dostępu dla właściciela, członków grupy oraz pozostałych użytkowników. W każdym z kręgów użytkownicy mają nadane uprawnienia do odczytu, zapisu i wykonywania, oznaczone odpowiednio literami *rwx*.
Do modyfikacji uprawnień wykorzystuję polecenia *chmod*.