**Informatyka - nowa formuła egzaminu dyplomowego (bez pracy dyplomowej) - pytania kierunkowe i specjalnościowe**

*1. Wyjaśnij, jak prawo autorskie chroni programy komputerowe oraz jakie są różne typy licencji oprogramowania. Omów konsekwencje prawne naruszenia tych licencji.*

**1. Ochrona programów komputerowych przez prawo autorskie i licencje**

Programy komputerowe są chronione jako utwory literackie w ramach prawa autorskiego. Oznacza to, że autorzy mają wyłączne prawa do kopi

owania, modyfikowania i dystrybucji kodu.  
**Typy licencji oprogramowania:**

* **Proprietary (zamknięte)** – np. Windows, Photoshop – użytkownik może korzystać, ale nie ma dostępu do kodu źródłowego.
* **Open Source (otwarte)** – np. Linux, Firefox – kod jest dostępny dla wszystkich, ale licencje określają warunki użytkowania (np. GPL, MIT).
* **Freeware, Shareware, Trial** – darmowe lub czasowo ograniczone wersje programów.  
  **Konsekwencje naruszeń:** Mogą obejmować kary finansowe, odpowiedzialność cywilną i karną oraz wykluczenie z rynku.

*2. Przedstaw kluczowe różnice między tradycyjnymi metodami zarządzania projektami (np. Waterfall) a metodykami zwinnymi (np. Scrum, Kanban) oraz omów, w jakich sytuacjach każda z tych metod może być najbardziej efektywna, szczególnie w kontekście projektów IT.*

**2. Różnice między Waterfall a Agile (Scrum, Kanban)**

* **Waterfall** – sztywna struktura, każda faza projektu (analiza, projektowanie, implementacja, testowanie) jest wykonywana po kolei. Dobra dla dużych projektów z jasno określonymi wymaganiami.
* **Agile (Scrum, Kanban)** – elastyczne podejście, iteracyjne, szybkie dostosowywanie do zmian. Scrum wykorzystuje sprinty, a Kanban pozwala na ciągły przepływ zadań.  
  **Kiedy stosować?**
* Waterfall – projekty rządowe, medyczne, gdzie wymagania są stałe.
* Agile – dynamiczne projekty IT, startupy, aplikacje webowe.

*3. Omów znacznie responsywności w wykorzystaniu systemów informatycznych.*

**3. Znaczenie responsywności w systemach informatycznych**

Responsywność oznacza szybkość reakcji systemu na działania użytkownika. Jest kluczowa dla UX (User Experience), wpływa na wydajność pracy i satysfakcję użytkowników. W systemach krytycznych, np. medycznych, może decydować o bezpieczeństwie pacjentów.

*4. Co to jest "Mobile First"w projektowaniu i dlaczego jest to ważne?*

**4. Mobile First – co to i dlaczego ważne?**

Podejście Mobile First oznacza projektowanie aplikacji/webów najpierw pod urządzenia mobilne, a potem dostosowywanie do większych ekranów. Jest to ważne, bo:

* Większość użytkowników korzysta z internetu na smartfonach.
* Wpływa na SEO – Google promuje strony przyjazne mobilnie.
* Poprawia UX, bo eliminuje zbędne elementy i optymalizuje działanie na mniejszych ekranach.

*5. Jak zorganizowane są dane w systemach plików i jakie to ma znaczenie z punktu widzenia bezpieczeństwa?*

**5. Organizacja danych w systemach plików a bezpieczeństwo**

Dane w systemach plików są organizowane hierarchicznie (katalogi, pliki). Kluczowe aspekty bezpieczeństwa:

* Uprawnienia dostępu (np. chmod w Linuxie).
* Szyfrowanie plików i dysków.
* Kontrola wersji i kopie zapasowe.
* Ograniczenia uprawnień dla użytkowników (zasada najmniejszych uprawnień).

*6. Podaj uwarunkowania wykorzystania systemów Content Management System (CMS).*

**6. Uwarunkowania wykorzystania CMS (Content Management System)**

CMS (np. WordPress, Joomla, Drupal) ułatwiają zarządzanie treścią, ale:

* Wymagają aktualizacji dla bezpieczeństwa.
* Mogą mieć ograniczenia w personalizacji (bez dodatkowego kodowania).
* Mogą być podatne na ataki (np. SQL Injection, ataki na wtyczki).

*7. Jakie mogą być najczęstsze zagrożenia dla bezpieczeństwa sieci teleinformatycznych?*

**7. Najczęstsze zagrożenia dla bezpieczeństwa sieci**

* **Ataki DDoS** – przeciążenie serwera.
* **Phishing** – wyłudzanie danych.
* **Malware, ransomware** – złośliwe oprogramowanie.
* **Man-in-the-middle** – przechwytywanie komunikacji.
* **Nieaktualne oprogramowanie** – podatności bezpieczeństwa.

*8. Czym są systemy wykrywania i zapobiegania intruzom (IDS/IPS) i jak one działają?*

**8. Systemy IDS/IPS – wykrywanie i zapobieganie intruzom**

* **IDS (Intrusion Detection System)** – monitoruje sieć i wykrywa potencjalne ataki, ale nie blokuje ich.
* **IPS (Intrusion Prevention System)** – działa aktywnie, wykrywa i blokuje ataki w czasie rzeczywistym.  
  Działają na zasadzie analizy sygnatur lub anomalii ruchu sieciowego.

*9. Jak działają zapory sieciowe (firewalle) i jakie mają ograniczenia?*

**9. Zapory sieciowe (firewalle) i ich ograniczenia**

Firewalle filtrują ruch sieciowy na podstawie reguł (np. IP, porty, protokoły). Mogą być sprzętowe lub programowe.  
**Ograniczenia:**

* Nie chronią przed atakami wewnętrznymi.
* Nie zabezpieczają przed phishingiem.
* Mogą blokować legalne usługi przy błędnej konfiguracji.

*10. Jakie są zasady bezpiecznego wykorzystywania systemów teleinformatycznych?*

**10. Zasady bezpiecznego wykorzystywania systemów teleinformatycznych**

* **Silne hasła** – unikanie prostych kombinacji, używanie menedżerów haseł.
* **Aktualizacje oprogramowania** – usuwanie luk w zabezpieczeniach.
* **Zasada najmniejszych uprawnień** – użytkownicy powinni mieć tylko niezbędne uprawnienia.
* **Kopia zapasowa (backup)** – ochrona przed awariami i ransomware.
* **Unikanie publicznych sieci Wi-Fi** – korzystanie z VPN w przypadku łączenia się w niezaufanych sieciach.
* **Edukacja użytkowników** – szkolenia z cyberbezpieczeństwa.

*11. W jaki sposób szyfrowanie wpływa na bezpieczeństwo systemów teleinformatycznych?*

**11. Wpływ szyfrowania na bezpieczeństwo systemów teleinformatycznych**

Szyfrowanie chroni dane przed nieautoryzowanym dostępem, poprzez konwersję ich do postaci nieczytelnej dla osób nieposiadających klucza deszyfrującego.  
**Przykłady:**

* **HTTPS** – szyfrowana komunikacja w sieci.
* **VPN** – tunelowanie ruchu internetowego.
* **AES, RSA** – algorytmy szyfrowania danych.

*12. Wymień i omów popierając przykładami główne zagrożenia w systemach teleinformatycznych*

**12. Główne zagrożenia w systemach teleinformatycznych**

* **Ataki DDoS** – przeciążenie serwerów np. blokowanie dostępu do strony rządowej.
* **Phishing** – fałszywe e-maile próbujące wyłudzić dane logowania.
* **Ransomware** – szyfrowanie plików i żądanie okupu (np. WannaCry).
* **SQL Injection** – atak na bazę danych poprzez wstrzyknięcie złośliwego kodu SQL.
* **Zero-day** – wykorzystanie niezałatanych luk bezpieczeństwa.

*13. Wymień i opisz co najmniej 4 usługi bezpieczeństwa informacji uwzględniając możliwe ataki na te usługi.*

**13. Usługi bezpieczeństwa informacji i możliwe ataki**

1. **Poufność** – ochrona przed nieautoryzowanym dostępem (zagrożenie: sniffing, ataki man-in-the-middle).
2. **Integralność** – zapewnienie, że dane nie zostały zmodyfikowane (zagrożenie: ataki typu man-in-the-middle).
3. **Dostępność** – zapewnienie dostępności usług (zagrożenie: ataki DDoS).
4. **Autoryzacja i uwierzytelnianie** – weryfikacja tożsamości użytkowników (zagrożenie: brute-force, kradzież sesji).

*14. Proszę wymienić i omówić sprzętowe oraz programowe składniki sieci komputerowych*

**14. Sprzętowe i programowe składniki sieci komputerowych**

**Sprzętowe:**

* **Router** – przekierowuje ruch w sieci.
* **Switch** – łączy urządzenia w sieci LAN.
* **Firewall sprzętowy** – filtruje ruch sieciowy.
* **Serwery** – przechowują dane i aplikacje.

**Programowe:**

* **Systemy operacyjne sieciowe** (np. Windows Server, Linux).
* **Oprogramowanie firewall** (np. iptables, pfSense).
* **VPN** – szyfrowanie ruchu internetowego.
* **Systemy IDS/IPS** – wykrywanie i zapobieganie atakom.

*15. Na czym polega efekt lawinowy w systemie kryptograficznym?*

**15. Efekt lawinowy w kryptografii**

Efekt lawinowy oznacza, że nawet minimalna zmiana w danych wejściowych (np. jednym bicie) powoduje znaczącą zmianę w wyniku szyfrowania. Jest kluczowy w algorytmach takich jak AES czy SHA-256, ponieważ zwiększa odporność na ataki.

*16. Proszę podać właściwości grafiki rastrowej i wektorowej.*

**16. Właściwości grafiki rastrowej i wektorowej**

* **Grafika rastrowa** (np. JPEG, PNG):
  + Składa się z pikseli.
  + Traci jakość po powiększeniu.
  + Nadaje się do zdjęć i realistycznych obrazów.
* **Grafika wektorowa** (np. SVG, AI):
  + Składa się z figur geometrycznych.
  + Można skalować bez utraty jakości.
  + Idealna do logotypów i ikon.

*17. Proszę podać i scharakteryzować znane topologie sieci komputerowych.*

**17. Znane topologie sieci komputerowych**

* **Magistrali** – wszystkie urządzenia podłączone do jednej linii (prosta, ale awaryjna).
* **Pierścienia** – dane krążą w jednym kierunku (wydajna, ale podatna na awarie).
* **Gwiazdy** – centralny punkt (switch/router) łączy wszystkie urządzenia (popularna i bezpieczna).
* **Siatki (mesh)** – każde urządzenie może łączyć się z każdym (wysoka redundancja).

*18. Omów zasady współpracy komputera z urządzeniami wejścia/wyjścia*

**18. Współpraca komputera z urządzeniami wejścia/wyjścia**

Komputer komunikuje się z urządzeniami poprzez sterowniki oraz interfejsy (USB, PCI, Bluetooth).  
**Przykłady:**

* Klawiatura i mysz (wejście) → przesyłanie sygnałów przez sterownik HID.
* Monitor (wyjście) → karta graficzna renderuje obraz.
* Drukarka (wyjście) → komputer wysyła dane w formacie obsługiwanym przez drukarkę (np. PCL, PostScript).

*19. Omów rodzaje testów i jakie korzyści przynosi automatyzacja testów w procesie wytwarzania oprogramowania?*

**19. Rodzaje testów i korzyści automatyzacji**

**Rodzaje testów:**

* **Jednostkowe** – testowanie pojedynczych modułów kodu.
* **Integracyjne** – sprawdzanie komunikacji między modułami.
* **Systemowe** – testowanie całości aplikacji.
* **Akceptacyjne** – testowanie zgodności z wymaganiami biznesowymi.

**Korzyści automatyzacji:**

* Szybsze wykrywanie błędów.
* Możliwość wielokrotnego powtarzania testów.
* Redukcja kosztów testowania.
* Mniejsze ryzyko błędu ludzkiego.

*20. Jakie funkcje oferują narzędzia do zarządzania projektami, które są kluczowe dla skutecznego zarządzania zespołem programistycznym?*

**20. Kluczowe funkcje narzędzi do zarządzania projektami**

Narzędzia takie jak Jira, Trello, Asana, Redmine oferują:

* **Zarządzanie zadaniami** – tworzenie i przypisywanie zadań.
* **Harmonogramowanie** – wykresy Gantta, sprinty Scrum.
* **Śledzenie postępów** – tablice Kanban.
* **Zarządzanie dokumentacją** – wiki, repozytoria kodu.
* **Integracje z narzędziami deweloperskimi** (np. GitHub, CI/CD).

*21. Jakie są najpopularniejsze systemy kontroli wersji i jakie są ich główne zalety i wady w pracy w dużych zespołach?*

**21. Najpopularniejsze systemy kontroli wersji i ich zalety/wady**

**Najpopularniejsze systemy:**

* **Git** (rozproszony) – używany w GitHub, GitLab, Bitbucket.
* **SVN (Subversion)** – scentralizowany, popularny w korporacjach.
* **Mercurial** – podobny do Gita, ale mniej popularny.

|  |  |
| --- | --- |
| **Zalety Gita dla dużych zespołów:** ✅ Rozproszony model – każdy ma pełną kopię repozytorium. ✅ Efektywność – szybkie operacje na lokalnym repozytorium. ✅ Branching – łatwe tworzenie i scalanie gałęzi. | **Wady:** ❌ Może być skomplikowany dla początkujących. ❌ Brak natywnej kontroli dostępu (trzeba używać GitHub/GitLab). |

*22. Czym są heurystyki Jacoba Nielsena? Omów jedną z nich.*

**22. Heurystyki Jacoba Nielsena – zasady użyteczności interfejsów**

Heurystyki Nielsena to 10 zasad UX, pomagające w projektowaniu intuicyjnych interfejsów.

**Przykład: "Zapewnij informację o stanie systemu"**

* Użytkownik powinien zawsze wiedzieć, co się dzieje (np. pasek ładowania, komunikaty błędów).
* Przykład: Gmail informuje o wysyłaniu wiadomości („Wiadomość została wysłana”).

*23. Jakie są główne etapy procesu UX produktu cyfrowego?*

**23. Etapy procesu UX produktu cyfrowego**

1. **Badania użytkowników** – analiza potrzeb, np. wywiady, ankiety.
2. **Tworzenie person i scenariuszy** – definiowanie grup docelowych.
3. **Wireframing i prototypowanie** – szkice i interaktywne modele UI.
4. **Testy użyteczności** – weryfikacja projektu na użytkownikach.
5. **Iteracyjne poprawki** – dostosowanie UX na podstawie testów.

*24. Jakie są główne kroki w procesie projektowania algorytmu i jak wpływają one na efektywność systemu informatycznego?*

**24. Główne kroki projektowania algorytmu i ich wpływ na system**

1. **Definicja problemu** – jasne określenie celu.
2. **Analiza danych wejściowych i wyjściowych** – określenie struktur danych.
3. **Projektowanie algorytmu** – wybór metody rozwiązania (np. rekurencja, programowanie dynamiczne).
4. **Analiza złożoności** – ocena wydajności (czas/pamięć).
5. **Implementacja i testowanie** – sprawdzenie poprawności działania.
6. **Optymalizacja** – ulepszanie algorytmu dla lepszej efektywności.

*25. W jaki sposób analiza złożoności obliczeniowej (czasowej i pamięciowej) wpływa na wybór algorytmu do konkretnego zadania?*

**25. Wpływ analizy złożoności obliczeniowej na wybór algorytmu**

* **Złożoność czasowa** określa, jak szybko rośnie czas wykonania w zależności od ilości danych.
  + np. **O(n log n)** (Merge Sort) jest lepsze niż **O(n²)** (Bubble Sort) dla dużych zbiorów.
* **Złożoność pamięciowa** – ile pamięci zajmuje algorytm.
  + np. DFS może być bardziej pamięciooszczędny niż BFS w niektórych sytuacjach.

*26. Jakie są najczęstsze techniki optymalizacji algorytmów i w jakich sytuacjach ich stosowanie jest najbardziej efektywne?*

**26. Najczęstsze techniki optymalizacji algorytmów i ich zastosowania**

* **Programowanie dynamiczne** – np. optymalizacja problemów plecakowych.
* **Podział i zwyciężaj (Divide & Conquer)** – np. QuickSort, MergeSort.
* **Kompresja danych** – np. Huffman coding w kompresji plików.
* **Równoległe przetwarzanie** – np. MapReduce w big data.
* **Przybliżone algorytmy** – np. heurystyki dla NP-trudnych problemów.

*27. Jak podejścia algorytmiczne mogą wpływać na skalowalność systemu informatycznego?*

**27. Wpływ podejść algorytmicznych na skalowalność systemu**

* **Efektywne algorytmy** (np. O(n log n) zamiast O(n²)) pozwalają na obsługę większej liczby użytkowników.
* **Struktury danych** (np. drzewa AVL zamiast list) mogą poprawić czas dostępu do danych.
* **Równoległość** – algorytmy równoległe (np. MapReduce) zwiększają skalowalność w systemach rozproszonych.

*28. Jakie są kluczowe różnice między podejściem algorytmicznym a heurystycznym w rozwiązywaniu problemów w systemach informatycznych?*

**28. Podejście algorytmiczne vs. heurystyczne w rozwiązywaniu problemów**

* **Podejście algorytmiczne** – ścisłe reguły, gwarantuje optymalne rozwiązanie (np. Dijkstra do wyznaczania najkrótszej ścieżki).
* **Podejście heurystyczne** – przybliżone rozwiązania, szybsze, ale nie zawsze optymalne (np. algorytm mrówkowy do problemu komiwojażera).

**Zastosowanie:**

* **Algorytmy** – lepsze w zadaniach wymagających precyzji (np. szyfrowanie).
* **Heurystyki** – lepsze w problemach NP-trudnych, gdy optymalne rozwiązanie jest zbyt kosztowne (np. AI, optymalizacja grafiki).

*29. Jak algebra liniowa może być wykorzystana w przetwarzaniu obrazów i analizie danych w systemach informatycznych?*

**29. Algebra liniowa w przetwarzaniu obrazów i analizie danych**

|  |  |
| --- | --- |
| **Przetwarzanie obrazów:**  Macierze do reprezentacji obrazów.  Transformaty (np. DCT w JPEG). | **Analiza danych:**  PCA (analiza głównych składowych) do redukcji wymiarów.  Sieci neuronowe – mnożenie macierzy w warstwach neuronowych. |

*30. W jaki sposób rachunek różniczkowy i całkowy przyczynia się do optymalizacji algorytmów i modeli predykcyjnych?*

**30. Rachunek różniczkowy i całkowy w optymalizacji algorytmów i modeli predykcyjnych**

* **Optymalizacja funkcji kosztu w AI** – gradient descent używa pochodnych do minimalizacji błędów w sieciach neuronowych.
* **Analiza szybkości zmian** – różniczkowanie pomaga w dynamicznym sterowaniu systemami.
* **Interpolacja i aproksymacja** – całkowanie pozwala modelować złożone funkcje w analizie danych.

*31. Jak teoria prawdopodobieństwa i statystyka wspierają podejmowanie decyzji w systemach rekomendacyjnych i analizie danych?*

**31. Teoria prawdopodobieństwa i statystyka w systemach rekomendacyjnych i analizie danych**

Systemy rekomendacyjne i analiza danych opierają się na modelach probabilistycznych i metodach statystycznych, aby przewidywać preferencje użytkowników oraz analizować wzorce w danych.

* **Systemy rekomendacyjne:**
  + **Filtracja kolaboratywna** – analiza podobieństw między użytkownikami (np. system Netflix przewiduje filmy na podstawie ocen innych użytkowników o podobnych gustach).
  + **Modele Bayesowskie** – prognozowanie na podstawie prawdopodobieństwa (np. filtry antyspamowe analizujące treść e-maili).
* **Analiza danych:**
  + **Regresja liniowa i wieloraka** – wykorzystywane do przewidywania trendów na podstawie wcześniejszych danych (np. prognozy sprzedaży).
  + **Analiza wariancji (ANOVA)** – stosowana do porównywania różnych grup danych (np. testowanie skuteczności reklam w różnych regionach).

*32. W jaki sposób logika matematyczna i teoria zbiorów mogą być wykorzystane w weryfikacji poprawności i niezawodności kodu źródłowego?*

**32. Logika matematyczna i teoria zbiorów w weryfikacji kodu**

Poprawność i niezawodność kodu źródłowego można analizować za pomocą metod formalnych bazujących na logice matematycznej i teorii zbiorów.

* **Logika matematyczna:**
  + **Model checking** – sprawdzanie, czy program spełnia określone własności (np. poprawność protokołów sieciowych).
  + **Twierdzenia Hoare’a** – analiza poprawności kodu na poziomie pre- i postwarunków.
* **Teoria zbiorów:**
  + **Optymalizacja zapytań SQL** – analiza operacji na zbiorach danych dla efektywniejszych wyników.
  + **Wykrywanie konfliktów typów danych** – stosowane w językach funkcyjnych (np. Haskell, Scala).

*33. Jak kryptografia, oparta na teoriach liczbowych i algebraicznych, wpływa na bezpieczeństwo systemów informatycznych?*

**33. Kryptografia oparta na teoriach liczbowych i algebraicznych**

Nowoczesne systemy kryptograficzne wykorzystują własności liczb pierwszych i operacje algebraiczne do zabezpieczania danych przed nieautoryzowanym dostępem.

* **Teoria liczb:**
  + **RSA** – oparty na trudności faktoryzacji dużych liczb pierwszych (stosowany w szyfrowaniu komunikacji).
  + **Krzywe eliptyczne** – bardziej efektywna metoda kryptograficzna używana w blockchainie i szyfrowaniu ECDSA.
* **Algebra abstrakcyjna:**
  + **Grupy i pierścienie** – wykorzystywane w kryptografii symetrycznej i asymetrycznej.
  + **Algorytm Diffie-Hellmana** – umożliwia bezpieczną wymianę kluczy w komunikacji (stosowany w HTTPS).

*34. Jak teoria grafów wpływa na projektowanie i implementację struktur danych takich jak drzewa, grafy i sieci?*

**34. Teoria grafów w projektowaniu struktur danych**

Teoria grafów znajduje zastosowanie w wielu aspektach informatyki, od organizacji danych po optymalizację wyszukiwania.

* **Drzewa:**
  + **Drzewa binarne (BST, AVL, B-drzewa)** – stosowane w bazach danych do szybkiego wyszukiwania.
* **Grafy:**
  + **Algorytm Dijkstry** – najkrótsza ścieżka w sieciach (np. GPS, nawigacja Google Maps).
  + **Algorytm PageRank** – analiza połączeń między stronami internetowymi (stosowany przez Google)

*35. W jaki sposób algebra liniowa jest stosowana w budowie i manipulacji strukturami danych, takimi jak macierze i wektory?*

**35. Algebra liniowa w strukturach danych**

Algebra liniowa jest kluczowa w reprezentacji i manipulacji danymi w postaci macierzy i wektorów.

* **Macierze:**
  + **Grafika komputerowa** – transformacje 3D w OpenGL.
  + **Sztuczna inteligencja** – operacje na zbiorach danych w uczeniu maszynowym.
* **Wektory:**
  + **Silniki rekomendacyjne** – Word2Vec w NLP do analizy znaczenia słów.
  + **Kompresja obrazów** – SVD stosowane w redukcji wymiarów.

*36. Jak zasady kombinatoryki są wykorzystywane w projektowaniu struktur danych optymalizujących wyszukiwanie i sortowanie?*

**36. Kombinatoryka w strukturach danych**

Techniki kombinatoryczne są wykorzystywane do optymalizacji operacji na strukturach danych.

* **Optymalizacja wyszukiwania:**
  + **Drzewa AVL** – minimalizacja wysokości drzewa dla szybszego wyszukiwania.
  + **Funkcje skrótu w tablicach haszujących** – minimalizacja kolizji.
* **Sortowanie:**
  + **Algorytm QuickSort** – optymalizacja sortowania poprzez podział i zwyciężaj.

*37. W jaki sposób teoria prawdopodobieństwa jest używana w strukturach danych takich jak filtry Blooma i drzewa probabilistyczne?*

**37. Teoria prawdopodobieństwa w filtrach Blooma i drzewach probabilistycznych**

Teoria prawdopodobieństwa pozwala na efektywne przechowywanie i wyszukiwanie danych z pewnym marginesem błędu.

* **Filtr Blooma:**
  + Używany w bazach danych (np. Google BigTable) do szybkiego sprawdzania obecności elementu w zbiorze.
* **Drzewa probabilistyczne:**
  + **Algorytm Huffmana** – kompresja danych poprzez drzewo kodowe.
  + **Modele Markowa** – przewidywanie kolejnych elementów (np. autokorekta w klawiaturach).

*38. Jak pojęcia z teorii liczb są stosowane w projektowaniu struktur danych dla kryptografii i zabezpieczeń?*

**38. Teoria liczb w strukturach danych dla kryptografii**

Zasady teorii liczb są stosowane w mechanizmach bezpieczeństwa.

* **Drzewa Merkle’a:**
  + Używane w blockchainie do weryfikacji integralności danych.
* **Generatory liczb losowych:**
  + Oparte na kongruencjach liniowych, np. w szyfrowaniu strumieniowym.

*39. Jakie są kluczowe etapy w procesie budowy modelu uczenia maszynowego i jak wpływają one na jakość i efektywność systemu sztucznej inteligencji?*

**39. Kluczowe etapy budowy modelu uczenia maszynowego**

1. **Zbieranie danych** – określenie jakości i źródła danych (np. dane medyczne, obrazy).
2. **Przetwarzanie danych** – czyszczenie i normalizacja, np. usuwanie duplikatów.
3. **Wybór modelu** – regresja, SVM, sieci neuronowe w zależności od problemu.
4. **Trenowanie modelu** – dostosowanie wag na podstawie zbioru treningowego.
5. **Ewaluacja modelu** – sprawdzanie jakości na zbiorze testowym.
6. **Optymalizacja hiperparametrów** – poprawianie wydajności (np. tuning learning rate).

*40. W jaki sposób techniki uczenia nadzorowanego i nienadzorowanego różnią się pod względem zastosowań i implementacji w systemach sztucznej inteligencji?*

**40. Uczenie nadzorowane vs. nienadzorowane**

* **Uczenie nadzorowane:**
  + Dane treningowe zawierają etykiety.
  + Zastosowanie: klasyfikacja (np. rozpoznawanie twarzy), regresja.
* **Uczenie nienadzorowane:**
  + Model sam znajduje wzorce w danych.
  + Zastosowanie: klasteryzacja (np. segmentacja klientów w e-commerce).

*41. Jakie są najlepsze praktyki w zakresie zarządzania danymi treningowymi, w tym ich gromadzenie, przechowywanie, przetwarzanie i zabezpieczanie, w kontekście budowy systemów sztucznej inteligencji?*

**41. Najlepsze praktyki w zarządzaniu danymi treningowymi w systemach sztucznej inteligencji**

Podczas budowy systemów AI kluczowe jest odpowiednie zarządzanie danymi treningowymi. Obejmuje to kilka aspektów:

* **Gromadzenie danych** – Dane powinny pochodzić z wiarygodnych i legalnych źródeł oraz być zgodne z regulacjami (np. RODO). Warto stosować techniki anonimizacji.
* **Przechowywanie** – Dane powinny być zabezpieczone przed nieautoryzowanym dostępem, a ich przechowywanie powinno uwzględniać redundancję (backupy, chmura).
* **Przetwarzanie** – Należy zadbać o jakość danych (czyszczenie, normalizacja, usuwanie błędów). Warto stosować techniki augmentacji danych w AI.
* **Zabezpieczanie** – Dane muszą być chronione przed wyciekiem (szyfrowanie, kontrola dostępu, audyty).

*42. Jak duże modele językowe (LLM) mogą być integrowane z istniejącymi systemami informatycznymi i aplikacjami, aby poprawić ich funkcjonalność i wydajność*

**42. Integracja dużych modeli językowych (LLM) z systemami IT**

Duże modele językowe, takie jak ChatGPT, można integrować z systemami IT na kilka sposobów:

* **Asystenci w aplikacjach** – LLM mogą wspierać obsługę klienta, generując automatyczne odpowiedzi w chatbotach.
* **Automatyczne generowanie treści** – Integracja z CMS pozwala na automatyczne tworzenie podsumowań, raportów czy e-maili.
* **Wyszukiwanie semantyczne** – Można je wykorzystać do analizy zapytań użytkowników i lepszego dopasowywania wyników w wyszukiwarkach.
* **Analiza danych** – LLM pomagają w przetwarzaniu dużych zbiorów danych i generowaniu raportów na ich podstawie.

*43. W jaki sposób duże modele językowe (LLM) mogą być wykorzystywane do automatyzacji zadań związanych z przetwarzaniem języka naturalnego (NLP), takich jak tłumaczenie, podsumowywanie tekstu i analiza sentymentu?*

**43. Wykorzystanie LLM w automatyzacji NLP**

LLM mogą automatyzować wiele zadań związanych z przetwarzaniem języka naturalnego:

* **Tłumaczenie tekstów** – Modele takie jak DeepL czy Google Translate wykorzystują AI do precyzyjniejszego tłumaczenia.
* **Podsumowywanie treści** – LLM mogą skracać długie artykuły, generując ich streszczenia.
* **Analiza sentymentu** – AI potrafi wykrywać emocje w komentarzach klientów, co jest używane np. w mediach społecznościowych i analizie opinii.

*44. Co jest przedmiotem praw autorskich, a co jest spod nich wyłączone? Przedstaw podział praw autorskich i omów je*

**44. Podział praw autorskich**

Prawa autorskie chronią utwory o charakterze twórczym, takie jak książki, programy komputerowe czy muzyka.

* **Co podlega ochronie?** – Teksty, grafiki, filmy, oprogramowanie, projekty graficzne.
* **Co nie podlega ochronie?** – Fakty, idee, procedury, teorie matematyczne.

Podział praw autorskich:

1. **Prawa osobiste** – niezbywalne (np. autorstwo utworu).
2. **Prawa majątkowe** – można nimi handlować (np. prawo do publikacji, sprzedaży).

*45. Wyjaśnij różnicę pomiędzy szyfrowaniem symetrycznym a asymetrycznym.*

**45. Szyfrowanie symetryczne vs asymetryczne**

* **Szyfrowanie symetryczne** – Używa jednego klucza do szyfrowania i deszyfrowania (np. AES).
  + Zalety: szybkie, wydajne.
  + Wady: problem z bezpiecznym przekazaniem klucza.
* **Szyfrowanie asymetryczne** – Używa dwóch kluczy: publicznego i prywatnego (np. RSA).
  + Zalety: bezpieczniejsze przy wymianie danych.
  + Wady: wolniejsze niż szyfrowanie symetryczne.

*46. Omówić przebieg bezpiecznej transmisji w sieciach teleinformatycznych na przykładzie protokołu SSL (Secure Socket Layer)*

**46. Bezpieczna transmisja w SSL**

SSL (obecnie TLS) zapewnia szyfrowanie danych przesyłanych przez internet.  
Przebieg transmisji:

1. Klient i serwer uzgadniają wersję protokołu i algorytmy szyfrowania.
2. Serwer przesyła certyfikat SSL.
3. Klient weryfikuje certyfikat i generuje klucz sesyjny.
4. Dane są szyfrowane i przesyłane bezpiecznym kanałem.

*47. Ataki odmowy dostępu do usług (Denial of Service) - podaj przykłady ataku i sposoby jego unikania.*

**47. Ataki Denial of Service (DoS) i sposoby ochrony**

Ataki DoS blokują dostęp do usług przez przeciążenie serwera.

Przykłady ataków:

* **Flooding** – wysyłanie ogromnej liczby żądań (np. SYN Flood).
* **DDoS** – atak z wielu komputerów jednocześnie.

Sposoby ochrony:

* Filtry ruchu (firewalle).
* Systemy wykrywania ataków IDS/IPS.
* Sieci CDN ograniczające wpływ ataków.

*48. Jaki jest cel, do czego jest używana maski podsieci w konfiguracji sieci?*

**48. Rola maski podsieci w konfiguracji sieci**

Maska podsieci określa, która część adresu IP jest siecią, a która urządzeniem.

* **Cel** – Dzieli sieć na mniejsze podsieci, zwiększając efektywność adresowania.
* **Przykład** – Maska **255.255.255.0** oznacza, że pierwsze trzy bajty IP to adres sieci, a ostatni to urządzenie.

*49. Wyjaśnij koncepcję wykorzystania Virtual Local Area Network (VLAN) i ich zalety w zarządzaniu siecią*

**49. VLAN i jego zalety**

VLAN (Virtual Local Area Network) to logiczny podział jednej fizycznej sieci na kilka podsieci.

* **Zalety VLAN:**
  + Lepsza kontrola ruchu i bezpieczeństwa.
  + Separacja sieci bez potrzeby dodatkowego sprzętu.
  + Redukcja ruchu broadcastowego.

*50. Jaką rolę odgrywają sztuczna inteligencja (AI) i uczenie maszynowe (ML) w nowoczesnych systemach wykrywania zagrożeń i jakie są wyzwania związane z ich wykorzystaniem?*

**50. AI i ML w wykrywaniu zagrożeń**

Sztuczna inteligencja pomaga wykrywać cyberzagrożenia poprzez analizę anomalii w ruchu sieciowym.

* **Zastosowania:**
  + Systemy wykrywania ataków (IDS).
  + Analiza logów pod kątem nietypowych zachowań.
  + Predykcja zagrożeń na podstawie wzorców.
* **Wyzwania:**
  + Fałszywe alarmy.
  + Duża ilość przetwarzanych danych wymaga mocy obliczeniowej.
  + Możliwość omijania algorytmów AI przez cyberprzestępców.

*51. Jakie są wady i zalety aplikacji natywnych w porównaniu do aplikacji internetowych w przypadku urządzeń mobilnych?*

**51. Wady i zalety aplikacji natywnych vs aplikacji internetowych na urządzeniach mobilnych**

Aplikacje **natywne** to programy stworzone specjalnie dla konkretnego systemu operacyjnego (np. Android, iOS), natomiast **aplikacje internetowe** działają w przeglądarce i nie wymagają instalacji.

✅ **Zalety aplikacji natywnych:**

* Lepsza wydajność i szybkość działania.
* Pełny dostęp do funkcji urządzenia (kamera, GPS, powiadomienia).
* Możliwość pracy offline.

❌ **Wady aplikacji natywnych:**

* Wyższy koszt i czas produkcji (osobne wersje na Androida i iOS).
* Konieczność aktualizacji przez użytkownika.

✅ **Zalety aplikacji internetowych:**

* Działają na wszystkich urządzeniach z przeglądarką.
* Aktualizacje są natychmiastowe i nie wymagają instalacji.

❌ **Wady aplikacji internetowych:**

* Wolniejsze działanie w porównaniu do natywnych.
* Ograniczony dostęp do funkcji urządzenia.
* Wymagają stałego połączenia z internetem.

*52. Wyjaśnij, czym jest wątek w kontekście systemów operacyjnych.*

**52. Wątek w systemach operacyjnych**

Wątek to podstawowa jednostka wykonawcza w systemie operacyjnym. Jest częścią procesu, ale ma własny licznik rozkazów i stos.

* **Wątki pozwalają na równoległe wykonywanie zadań**, np. odtwarzanie muzyki i przeglądanie internetu jednocześnie.
* **Zalety wielowątkowości**: poprawa wydajności, efektywne wykorzystanie procesora, szybsza reakcja aplikacji.
* **Problemy**: synchronizacja wątków (np. dostęp do tych samych zasobów).

*53. Czym jest wirtualizacja w kontekście systemów operacyjnych i jakie przynosi korzyści?*

**53. Wirtualizacja w systemach operacyjnych**

Wirtualizacja to technika, która pozwala uruchamiać wiele systemów operacyjnych na jednym fizycznym komputerze.

✅ **Korzyści:**

* Efektywne wykorzystanie sprzętu.
* Możliwość testowania różnych systemów operacyjnych.
* Łatwe skalowanie infrastruktury IT.

Przykłady narzędzi: VMware, VirtualBox, Hyper-V.

*54. Jakie są metody zapewniania bezpieczeństwa w systemach operacyjnych?*

**54. Metody zapewniania bezpieczeństwa w systemach operacyjnych**

* **Kontrola dostępu** – systemy uprawnień do plików i katalogów.
* **Szyfrowanie danych** – zabezpieczenie plików przed nieautoryzowanym dostępem.
* **Firewalle i antywirusy** – ochrona przed atakami sieciowymi.
* **Sandboxing** – uruchamianie aplikacji w izolowanym środowisku.
* **Regularne aktualizacje** – łatanie luk w zabezpieczeniach.

*55. Czym jest system plików i jakie są jego podstawowe funkcje?*

**55. System plików i jego funkcje**

System plików organizuje dane na nośniku (np. dysku twardym, SSD).

🔹 **Podstawowe funkcje:**

* Przechowywanie i organizacja plików.
* Zarządzanie dostępem do danych.
* Zapewnienie integralności danych.

Przykłady systemów plików: NTFS, FAT32, EXT4.

*56. Czym jest hypervisor w wirtualizacji, jaką pełni rolę i jak jest zorganizowany?*

**56. Hypervisor – rola i organizacja**

Hypervisor to oprogramowanie zarządzające maszynami wirtualnymi.

🔹 **Rodzaje hypervisorów:**

* **Typ 1 (bare-metal)** – działa bezpośrednio na sprzęcie (np. VMware ESXi).
* **Typ 2 (hostowane)** – działa jako program w systemie operacyjnym (np. VirtualBox).

✅ **Zalety wirtualizacji:** izolacja systemów, optymalizacja zasobów, łatwe tworzenie kopii zapasowych.

*57. Omów strukturę rejestru systemu operacyjnego?*

**57. Struktura rejestru systemu operacyjnego**

Rejestr w Windows to baza danych przechowująca ustawienia systemu i aplikacji.

🔹 **Główne gałęzie rejestru:**

* **HKEY\_CLASSES\_ROOT** – informacje o typach plików i rozszerzeniach.
* **HKEY\_CURRENT\_USER** – ustawienia użytkownika.
* **HKEY\_LOCAL\_MACHINE** – globalne ustawienia systemu.
* **HKEY\_USERS** – profile użytkowników.
* **HKEY\_CURRENT\_CONFIG** – informacje o aktualnej konfiguracji sprzętowej.

*58. Omów mechanizmy zapewniania bezpieczeństwa i niezawodności w systemach zarządzania bazami danych.*

**58. Mechanizmy bezpieczeństwa w bazach danych**

* **Kontrola dostępu** – użytkownicy mają różne poziomy uprawnień.
* **Szyfrowanie danych** – np. TLS w połączeniach do bazy.
* **Backupy** – ochrona przed utratą danych.
* **Replikacja** – zwiększenie niezawodności bazy.
* **ACID** – zasady zapewniające integralność danych.

*59. Wyjaśnij co oznacza i dlaczego mówimy o systemach operacyjnych 32 i 64 bitowych?*

**59. Systemy operacyjne 32-bitowe vs. 64-bitowe**

* **32-bitowy system** obsługuje do **4 GB RAM**, ograniczona wydajność.
* **64-bitowy system** obsługuje **więcej pamięci** i może uruchamiać zarówno aplikacje 32-, jak i 64-bitowe.

✅ **Zalety 64-bitowego systemu:** lepsza wydajność, możliwość obsługi większej ilości pamięci RAM, lepsza obsługa aplikacji wymagających dużej mocy obliczeniowej.

*60. Jakie są różnice i podobieństwa w sposobach deklaracji, definicji oraz wykorzystania zmiennych typów prostych w językach C, Java i C#?*

**60. Deklaracja, definicja i wykorzystanie zmiennych w C, Java i C#**

🔹 **C**

* **Deklaracja:** int a; (tylko informacja o typie).
* **Definicja:** int a = 5; (tworzenie zmiennej i przypisanie wartości).
* **Wskaźniki:** int \*ptr; (przechowuje adres innej zmiennej).

🔹 **Java**

* **Deklaracja i definicja:** int a = 5; (w Javie zmienne muszą należeć do klasy/metody).
* **Typy referencyjne:** obiekty przechowywane na stercie, np. String s = "Hello";.

🔹 **C#**

* **Podobne do Javy, ale obsługuje wskaźniki w trybie unsafe:**

unsafe {

int\* ptr;

}

* **Typy wartościowe i referencyjne:** int (przechowywany na stosie), string (na stercie).

✅ **Podobieństwa:** składnia deklaracji zmiennych jest zbliżona.  
❌ **Różnice:** zarządzanie pamięcią, obsługa wskaźników (C pozwala na bezpośredni dostęp, Java i C# działają na GC).

*61. W jaki sposób realizowana jest obsługa wyjątków w wybranym języku programowania i jakie są jej główne mechanizmy oraz najlepsze praktyki?*

**61. Obsługa wyjątków w wybranym języku programowania – mechanizmy i najlepsze praktyki**

Obsługa wyjątków jest mechanizmem pozwalającym na zarządzanie błędami w programie. Dzięki niej, kiedy pojawia się błąd, program nie przerywa działania, lecz reaguje na niego w zaplanowany sposób. W językach takich jak Java, C# czy Python, wyjątkami zarządza się przy pomocy bloków try-catch. Blok try zawiera kod, który może spowodować wyjątek, a blok catch obsługuje go, oferując odpowiednią reakcję.

**Najlepsze praktyki:**

* Zawsze obsługuj konkretne wyjątki, unikaj ogólnych catch.
* Unikaj używania wyjątków do kontrolowania przepływu programu (np. zamiast try-catch do logiki).
* Korzystaj z finally do zwalniania zasobów, np. zamykania plików.

*62. Omów, dla jakich danych zastosujesz tablicę. Podaj przykłady z deklaracją, utworzeniem i zainicjowaniem tablicy w języku C# lub Java.*

**62. Tablica – zastosowanie i przykłady w C# oraz Java**

Tablica to struktura danych, która pozwala na przechowywanie wielu wartości tego samego typu w jednym obiekcie. Zaletą tablicy jest szybki dostęp do elementów za pomocą indeksu, jednak rozmiar tablicy jest stały po jej zadeklarowaniu.

**Przykład w C#:**

int[] liczby = new int[5] {1, 2, 3, 4, 5};

**Przykład w Java:**

int[] liczby = {1, 2, 3, 4, 5};

**Zastosowanie:** Tablice są używane np. do przechowywania wyników kalkulacji, zbiorów danych czy elementów w grze komputerowej.

*63. Jak zdefiniować upstream branch, a następnie skonfigurować go w zdalnym repozytorium tak, aby miał tę samą nazwę co w repozytorium lokalnym oraz jak skonfigurować go pod inną nazwą niż w repozytorium lokalnym?*

**63. Konfiguracja upstream branch w Git**

Upstream branch w systemie kontroli wersji Git to gałąź, która jest połączona z zdalnym repozytorium, na przykład z serwerem GitHub. Aby skonfigurować upstream branch, używamy komendy:

git push --set-upstream origin nazwa\_gałęzi

Można również nadać inną nazwę gałęzi w repozytorium zdalnym:

git push --set-upstream origin lokalna\_gałąź:inna\_nazwa\_zdalna

To zapewnia, że gałąź w repozytorium lokalnym jest powiązana z odpowiednią gałęzią zdalną.

*64. Jak zdefiniować pojęcia tip i head w kontekście systemów kontroli wersji, oraz w jaki sposób można określić liczbę heads w repozytorium?*

**64. Tip i Head w systemach kontroli wersji**

W kontekście systemów kontroli wersji, tip oznacza najnowszą zmianę w danej gałęzi, czyli najbardziej aktualny commit. Head to wskaźnik na aktualny commit w gałęzi roboczej. Można również mówić o HEAD, które wskazuje na ostatni commit w aktualnie wybranej gałęzi.

Aby sprawdzić liczbę heads w repozytorium, należy użyć komendy:

git show-ref –heads

*65. Jakie są kluczowe różnice między klasą a obiektem w programowaniu obiektowym, oraz jak te różnice wpływają na tworzenie i wykorzystanie kodu?*

**65. Klasa vs obiekt w programowaniu obiektowym**

* **Klasa** to definicja szablonu dla obiektów, zawierająca zmienne i metody. Jest jak plan budowy domu, natomiast obiekt jest jego fizycznym odpowiednikiem.
* **Obiekt** to instancja klasy, tworzona na podstawie tej definicji, która zawiera konkretne dane.

**Przykład w Java:**

class Samochod {

String model;

void uruchomSilnik() {

System.out.println("Silnik uruchomiony");

}

}

Samochod auto = new Samochod();

W tym przykładzie Samochod to klasa, a auto to obiekt.

*66. Czym jest enkapsulacja w programowaniu obiektowym, jakie są jej główne zalety i dlaczego jest istotna dla tworzenia solidnego i bezpiecznego kodu?*

**66. Enkapsulacja – znaczenie i zalety**

Enkapsulacja to technika w programowaniu obiektowym polegająca na ukrywaniu szczegółów implementacji klasy i udostępnianiu tylko niezbędnych metod lub zmiennych do interakcji z obiektem. Dzięki temu kontrolujemy dostęp do danych i metod, co zapewnia bezpieczeństwo i łatwiejsze zarządzanie kodem.

**Zalety:**

* Ochrona danych przed nieautoryzowanym dostępem.
* Łatwiejsze utrzymanie kodu, ponieważ zmiany wewnętrzne w klasie nie wpływają na jej zewnętrzne wykorzystanie.

**Przykład w Java:**

class BankAccount {

private double saldo;

public void deposit(double amount) {

if (amount > 0) saldo += amount;

}

public double getSaldo() {

return saldo;

}

}

*67. Jakie jest znaczenie polimorfizmu w programowaniu obiektowym i jakie są przykłady jego praktycznego zastosowania w różnych językach programowania?*

**67. Polimorfizm – znaczenie i przykłady**

Polimorfizm to cecha programowania obiektowego, która pozwala na wykorzystanie tej samej metody w różnych klasach. Pozwala to na elastyczność i ponowne użycie kodu.

**Przykład w Java:**

class Zwierze {

void dzwiek() {

System.out.println("Dźwięk zwierzęcia");

}

}

class Kot extends Zwierze {

void dzwiek() {

System.out.println("Miau");

}

}

Polimorfizm pozwala na to, by obiekt Kot zachowywał się zgodnie z własną implementacją metody dzwiek(), nawet jeśli jego typ to Zwierze.

*68. Jakie są różnice między modyfikatorami dostępu public, private i protected w różnych językach programowania oraz jak te różnice wpływają na sposób zarządzania dostępem do członków klasy?*

**68. Modyfikatory dostępu w różnych językach programowania**

Modyfikatory dostępu kontrolują, który kod może uzyskać dostęp do członków klasy. W różnych językach (np. Java, C#) są stosowane modyfikatory takie jak:

* **public** – dostęp do członka klasy z każdej lokalizacji w kodzie.
* **private** – dostęp tylko wewnątrz klasy.
* **protected** – dostęp w obrębie klasy i klas dziedziczących.

Przykład w Java:

public class Osoba {

private String imie;

protected int wiek;

}

*69. Wyjaśnij pojęcie klasy i obiektu. Definiowanie klasy, tworzenie i usuwanie obiektów w wybranym języku programowania.*

**69. Klasa i obiekt – definicja, tworzenie i usuwanie**

* **Klasa** jest definicją, która wskazuje, jakie właściwości i metody będą dostępne dla tworzonych obiektów.
* **Obiekt** to instancja klasy, która może być stworzona za pomocą słowa kluczowego new.

**Przykład w Java:**

class Samochod {

String model;

}

Samochod auto = new Samochod();

Usuwanie obiektów w Java odbywa się automatycznie przez **Garbage Collector**, który zarządza pamięcią.

*70. Omów składnię definiowania oraz różnice w przekazywaniu parametrów, zwracaniu wartości i wywoływaniu metod obiektowych (instancyjnych) i klasowych (statycznych) na przykładzie wybranego języka.*

**70. Metody instancyjne vs statyczne w C#**

* **Metody instancyjne** są wywoływane na obiektach i mogą korzystać z instancji klasy.
* **Metody statyczne** są wywoływane na samej klasie, nie wymagają tworzenia obiektów.

**Przykład w C#:**

class Matematyka {

public static int Dodaj(int a, int b) { return a + b; }

}

Matematyka.Dodaj(3, 4); // Wywołanie metody statycznej

class Samochod {

public void Uruchom() { Console.WriteLine("Silnik uruchomiony!"); }

}

Samochod auto = new Samochod();

auto.Uruchom(); // Wywołanie metody instancyjnej

Metody statyczne są używane, gdy nie ma potrzeby tworzenia obiektów, np. w funkcjach pomocniczych.

*71. Opisz ideę dziedziczenia i jej realizację w wybranym obiektowym języku programowania.*

**71. Ideologia dziedziczenia w programowaniu obiektowym**

Dziedziczenie to jedna z podstawowych zasad programowania obiektowego, która umożliwia tworzenie nowych klas na podstawie już istniejących, odziedziczając ich właściwości i metody. Pozwala to na organizowanie klas w hierarchię, gdzie klasy pochodne (subklasy) dziedziczą po klasach bazowych (superklasach). Dzięki dziedziczeniu możemy ponownie wykorzystać kod, a także rozszerzać funkcjonalność bez konieczności pisania tego samego kodu od nowa.

**Przykład w Javie:**

class Pojazd {

void uruchomSilnik() {

System.out.println("Silnik uruchomiony!");

}

}

class Samochod extends Pojazd {

void włączKlimatyzację() {

System.out.println("Klimatyzacja włączona!");

}

}

public class Main {

public static void main(String[] args) {

Samochod samochod = new Samochod();

samochod.uruchomSilnik(); // metoda odziedziczona

samochod.włączKlimatyzację(); // metoda specyficzna dla Samochodu

}

}

W tym przypadku klasa Samochod dziedziczy metodę uruchomSilnik z klasy Pojazd.

*72. Jakie są kluczowe różnice między zmiennymi automatycznymi a dynamicznymi w kontekście programowania, oraz jak te różnice wpływają na zarządzanie pamięcią i działanie programu?*

**72. Zmienne automatyczne a dynamiczne – różnice i wpływ na pamięć**

**Zmienne automatyczne** są tworzone i usuwane w czasie działania funkcji, w której zostały zadeklarowane. Przechowywane są na stosie, co zapewnia szybki dostęp, ale mają ograniczony czas życia. Są automatycznie usuwane, gdy funkcja kończy działanie.

**Zmienne dynamiczne** są tworzone na stercie (heap), co daje im możliwość dłuższego życia niż zmienne automatyczne. Muszą być zarządzane ręcznie lub przez system garbage collection (np. w Javie czy Pythonie).

**Zalety i wady:**

* **Zmienne automatyczne:** Szybki dostęp, ale ograniczony czas życia i wielkość.
* **Zmienne dynamiczne:** Większa elastyczność, ale wymagają zarządzania pamięcią, co może prowadzić do wycieków pamięci.

*73. Jak różni się pojęcie interfejsu od implementacji klasy w kontekście programowania obiektowego, oraz jakie są główne zalety korzystania z interfejsów w projektowaniu oprogramowania?*

**73. Interfejs vs implementacja klasy – zalety interfejsów**

W programowaniu obiektowym, **interfejs** to kontrakt, który definiuje zestaw metod, które muszą zostać zaimplementowane przez klasy go implementujące. Interfejsy nie zawierają implementacji metod, jedynie ich deklaracje.

**Implementacja klasy** to z kolei rzeczywista definicja metod i zmiennych w klasie. Klasa implementuje interfejs, ale może zawierać także dodatkową funkcjonalność, która nie została określona w interfejsie.

**Zalety interfejsów:**

* Zwiększają elastyczność i umożliwiają wielokrotne dziedziczenie metod w językach, które tego nie wspierają (np. Java).
* Ułatwiają tworzenie testowalnych i łatwych do utrzymania systemów, ponieważ można je wymieniać, nie zmieniając implementacji.

*74. Jakie są główne techniki i zalety tworzenia nowych klas poprzez dziedziczenie z już istniejących klas w programowaniu obiektowym, oraz w jaki sposób dziedziczenie wspomaga reużywalność kodu i organizację hierarchii klas?*

**74. Dziedziczenie klas – techniki i zalety**

Tworzenie nowych klas poprzez **dziedziczenie** z istniejących klas pozwala na reużywanie kodu, co upraszcza organizację programu. Dziedziczenie wspomaga tworzenie hierarchii klas, w których klasy dziedziczą właściwości i metody innych klas.

**Techniki:**

* **Dziedziczenie pojedyncze:** Klasa pochodna dziedziczy po jednej klasie bazowej.
* **Dziedziczenie wielokrotne (w językach wspierających):** Klasa pochodna dziedziczy po wielu klasach.

**Zalety:**

* Redukcja powtarzalności kodu.
* Ułatwienie rozwoju programu przez dodawanie nowych funkcji bez zmiany istniejącego kodu.

*75. Podaj dwa przykłady programów komputerowych, które są wykorzystywane do symulacji komputerowej, oraz omów ich zastosowanie i główne cechy, które sprawiają, że są użyteczne w symulacji.*

**75. Przykłady programów do symulacji komputerowej**

1. **MATLAB/Simulink:**
   * **Zastosowanie:** Używany w inżynierii do modelowania, symulacji i analizy systemów dynamicznych.
   * **Główne cechy:** Obsługuje matematyczne i statystyczne obliczenia, oferuje szeroki zestaw bibliotek do różnych dziedzin.
2. **ANSYS:**
   * **Zastosowanie:** Symulacja i analiza inżynierska, szczególnie w zakresie mechaniki ciał stałych, przepływów i innych.
   * **Główne cechy:** Silne narzędzia do analizy strukturalnej, cieplnej, elektromagnetycznej.

*76. Model tworzony w Scilabie/Matlabie jest przybliżeniem rzeczywistości. Dokładnemu odwzorowaniu towarzyszą niepewności np. co do -parametrów modelu -warunków brzegowych które mogą wpływać na dokładność i wiarygodność wyników. Omów te niepewności na dowolnym przykładzie.*

**76. Niepewności w modelach matematycznych (Scilab/Matlab)**

Model tworzony w narzędziach takich jak Scilab czy Matlab jest tylko przybliżeniem rzeczywistości. Zawsze występują pewne **niepewności**, np. w parametrach modelu (np. współczynniki oporu w modelu aerodynamicznym) lub warunkach brzegowych (np. zmienne temperatury w modelu cieplnym). Te niepewności mogą wpływać na dokładność wyników i ich wiarygodność, dlatego często stosuje się **analizę wrażliwości**, aby zrozumieć, jak zmiany w danych wejściowych wpływają na wyniki.

**Przykład:**

* Modelowanie przepływu powietrza w turbinie wiatrowej może być obarczone niepewnością w określaniu prędkości wiatru, co wpływa na dokładność prognozowanej energii.

*77. Jak Python zarządza pamięcią dla obiektów i jakie są główne mechanizmy oraz strategie, które są stosowane w tym języku programowania?*

**77. Zarządzanie pamięcią w Pythonie**

Python zarządza pamięcią za pomocą **automatycznego zarządzania pamięcią** i **garbage collection**. Zmienna w Pythonie nie jest przypisana do konkretnego miejsca w pamięci, a obiekt, który jej odpowiada, jest przechowywany w dynamicznie alokowanej pamięci (sterta).

**Mechanizmy:**

* **Garbage collection:** Python automatycznie zwalnia pamięć, gdy obiekt nie jest już używany.
* **Referencje:** Liczba referencji do obiektów jest monitorowana, a obiekt zostaje usunięty, gdy nie ma do niego żadnej referencji.

*78. Wyjaśnij, co to jest wiązanie danych (Data Binding) w kontekście Windows Presentation Foundation i jak może być używane do synchronizacji danych między modelem danych a interfejsem użytkownika.*

**78. Wiązanie danych (Data Binding) w WPF**

**Data Binding** w WPF to mechanizm, który pozwala na synchronizację danych między modelem danych a interfejsem użytkownika. Główna idea polega na tym, że zmiany w danych modelu automatycznie odzwierciedlają się w interfejsie, a zmiany w interfejsie mogą być przekazywane do modelu.

**Przykład w XAML i C#:**

<TextBox Text="{Binding Name}" />

W tym przypadku Text w kontrolce TextBox jest powiązany z właściwością Name w modelu danych. Zmiana w Name zaktualizuje tekst w polu.

*79. Wyjaśnij, jak Extensible Application Markup Language współpracuje z kodem C# w aplikacjach Windows Presentation Foundation. Podaj przykład, w jaki sposób można zdefiniować interfejs użytkownika i manipulować nim z poziomu kodu C#.*

**79. WPF i XAML w aplikacjach C#**

WPF (Windows Presentation Foundation) wykorzystuje **XAML** (Extensible Application Markup Language) do definiowania interfejsu użytkownika. XAML to deklaratywny język znaczników, który pozwala na opisanie struktury UI w sposób wizualny.

**Przykład:**

<Button Content="Kliknij mnie" Click="OnButtonClick"/>

Tutaj definiujemy przycisk w XAML, a jego logikę obsługujemy w C# w metodzie OnButtonClick.

*80. Omów zasady budowy interfejsów użytkownika systemów informatycznych.*

**80. Budowa interfejsów użytkownika systemów informatycznych**

Zasady budowy interfejsów użytkownika obejmują tworzenie intuicyjnych, efektywnych i przyjaznych dla użytkownika UI. Dobre UI powinno być **spójne**, **czytelne** i **dostosowane do potrzeb użytkownika**.

**Zasady:**

* **Zrozumiałość:** Prosty i logiczny układ elementów.
* **Responsywność:** Interfejs dostosowuje się do różnych urządzeń.
* **Intuicyjność:** Użytkownik łatwo rozumie, jak korzystać z systemu.

*81. Omów proces normalizacji baz i jej trzy postaci popierając przykładami.*

**81. Proces normalizacji baz danych i jej trzy postaci**

**Normalizacja** to proces organizowania danych w bazach danych w celu eliminacji redundancji i zapewnienia integralności danych. Głównym celem normalizacji jest zmniejszenie duplikacji danych, co poprawia efektywność przechowywania i aktualizowania informacji.

* **Pierwsza postać normalna (1NF):** Zakłada, że każda kolumna w tabeli zawiera tylko pojedyncze wartości (atomowe) oraz, że wszystkie rekordy są unikalne. Dodatkowo, wszystkie atrybuty muszą być jednorodne, tzn. zawierać dane tego samego typu.

**Przykład (1NF):** Przed normalizacją:

ID | Imię | Zajęcia

1 | Jan | Matematyka, Fizyka

2 | Anna | Historia

Po normalizacji do 1NF:

ID | Imię | Zajęcia

1 | Jan | Matematyka

1 | Jan | Fizyka

2 | Anna | Historia

**Druga postać normalna (2NF):** Wymaga, aby spełniona była 1NF, a także, aby każda kolumna była zależna od całkowitej klucza głównego, a nie tylko od jego części.

**Przykład (2NF):** Przed normalizacją:

ID | Nazwa kursu | Imię nauczyciela | Zajęcia

1 | Matematyka | Jan Kowalski | Fizyka

1 | Fizyka | Anna Nowak | Matematyka

Po normalizacji do 2NF:

Tabela 1 (Kursy): ID | Nazwa kursu

1 | Matematyka

2 | Fizyka

Tabela 2 (Nauczyciele): Kurs\_ID | Nauczyciel

1 | Jan Kowalski

2 | Anna Nowak

**Trzecia postać normalna (3NF):** Wymaga, aby spełniona była 2NF, a także aby każda kolumna niekluczowa była zależna bezpośrednio od klucza głównego i nie miała zależności przechodnich (tzn. nie może być zależna od innych kolumn, które nie są kluczem głównym).

**Przykład (3NF):** Przed normalizacją:

ID | Imię | Miasto | Kod pocztowy

1 | Jan | Warszawa | 00-000

2 | Anna | Kraków | 30-000

Po normalizacji do 3NF:

Tabela 1 (Osoby): ID | Imię | Miasto\_ID

1 | Jan | 1

2 | Anna | 2

Tabela 2 (Miasta): Miasto\_ID | Miasto | Kod pocztowy

1 | Warszawa | 00-000

2 | Kraków | 30-000

*82. W jaki sposób notacja BPMN (ang. Business Process Model and Notation) może być wykorzystana do modelowania złożonych procesów biznesowych w organizacji, i jakie wyzwania mogą się pojawić podczas jej implementacji w systemach informatycznych?*

**82. Notacja BPMN i wyzwania w implementacji**

**BPMN (Business Process Model and Notation)** to standard notacji graficznej służący do modelowania procesów biznesowych. Dzięki BPMN można tworzyć zrozumiałe i spójne diagramy procesów, które pomagają w analizie, automatyzacji i optymalizacji działań organizacyjnych.

**Zastosowanie BPMN:**

* **Wizualizacja procesów:** BPMN umożliwia tworzenie przejrzystych i zrozumiałych schematów procesów, co ułatwia zrozumienie skomplikowanych działań w organizacji.
* **Komunikacja z interesariuszami:** Umożliwia przekazywanie zrozumiałych informacji między zespołami biznesowymi i technicznymi.
* **Optymalizacja:** Pozwala na identyfikację wąskich gardeł, redundancji i innych problemów w procesach.

**Wyzwania:**

* **Złożoność procesów:** Duża liczba elementów BPMN może sprawić, że diagramy staną się nieczytelne.
* **Integracja z systemami IT:** Implementacja procesów BPMN w systemach informatycznych może wymagać zaawansowanych narzędzi do automatyzacji procesów.
* **Szkolenie zespołów:** Użycie BPMN wymaga pewnego poziomu wiedzy i doświadczenia, co może stanowić wyzwanie w organizacjach bez odpowiedniego przeszkolenia.

*83. Proszę omówić i porównać modele relacyjne i nierelacyjne bazy danych.*

**83. Modele relacyjne vs. nierelacyjne bazy danych**

**Modele relacyjne:** Bazują na tabelach, gdzie dane są przechowywane w wierszach i kolumnach. Relacyjne bazy danych, jak MySQL, PostgreSQL, SQL Server, używają języka SQL do zarządzania danymi.

**Zalety:**

* Struktura danych jest bardzo uporządkowana i spójna.
* Zastosowanie kluczy głównych i obcych zapewnia integralność danych.
* Obsługują transakcje ACID (Atomicity, Consistency, Isolation, Durability).

**Wady:**

* Skalowalność może być problemem w przypadku dużych zbiorów danych.
* Modele relacyjne mogą być mniej elastyczne w przypadku nieustrukturalizowanych danych.

**Modele nierelacyjne (NoSQL):** Bazy danych, które nie stosują tabeli i wierszy. Zamiast tego mogą używać różnych formatów przechowywania danych, takich jak dokumenty, klucze-wartości, grafy czy kolumny.

**Zalety:**

* Lepsza skalowalność pozioma.
* Elastyczność w przechowywaniu danych niestrukturalnych.
* Wydajność w obsłudze dużych ilości danych w czasie rzeczywistym.

**Wady:**

* Brak jednolitej struktury może prowadzić do problemów z integralnością danych.
* Brak standardowego języka zapytań.

*84. Proszę omówić zasady definiowania wzorców funkcji i klas (znane również jako szablony lub generics).*

**84. Definiowanie wzorców funkcji i klas (Generics)**

**Generics** (wzorce szablonów) to technika umożliwiająca tworzenie funkcji i klas, które mogą działać na różnych typach danych bez potrzeby wielokrotnego definiowania tych samych funkcji. Generics zwiększają reużywalność i bezpieczeństwo kodu, ponieważ umożliwiają operowanie na typach w sposób ogólny i elastyczny.

**Przykład w Javie (Generics):**

class Para<T, U> {

private T pierwszy;

private U drugi;

public Para(T pierwszy, U drugi) {

this.pierwszy = pierwszy;

this.drugi = drugi;

}

public T getPierwszy() {

return pierwszy;

}

public U getDrugi() {

return drugi;

}

}

Tutaj Para<T, U> to klasa, która może przechowywać dwa obiekty różnych typów, np. Para<Integer, String>.

*85. Proszę podać główne cechy zunifikowanego języka modelowania systemów informatycznych (ang. UML ) – zastosowanie, charakterystykę oraz jego podstawowe diagramy.*

**85. Zunifikowany język modelowania systemów informatycznych (UML)**

**UML (Unified Modeling Language)** to język służący do tworzenia modeli systemów informatycznych, umożliwiający ich graficzne przedstawienie i zrozumienie.

**Główne cechy:**

* **Zastosowanie:** UML jest szeroko stosowany w inżynierii oprogramowania do projektowania, dokumentowania i analizowania systemów.
* **Diagramy:** UML oferuje różne typy diagramów, takie jak:
  + **Diagramy klas:** Przedstawiają struktury obiektów w systemie.
  + **Diagramy przypadków użycia:** Pokazują interakcje między użytkownikami a systemem.
  + **Diagramy sekwencji:** Ilustrują kolejność komunikatów między obiektami.

*86. Omów pojęcie abstrakcji i wskaż jego znaczenie z punktu widzenia wielokrotnego użycia (ang. Reuse).*

**86. Abstrakcja i jej znaczenie dla wielokrotnego użycia (Reuse)**

**Abstrakcja** to proces ukrywania szczegółów implementacji i skupianie się na istotnych właściwościach obiektu lub systemu. Pozwala to na **wielokrotne użycie** komponentów i klas w różnych częściach systemu, bez konieczności ponownego pisania kodu.

**Znaczenie:**

* Umożliwia tworzenie uniwersalnych rozwiązań, które mogą być wykorzystywane w różnych kontekstach.
* Ułatwia rozwój systemów przez oddzielanie logiki biznesowej od szczegółów implementacyjnych.

*87. Mechanizmy zapewniania bezpieczeństwa i niezawodności w systemach zarządzania bazami danych.*

**87. Mechanizmy zapewniania bezpieczeństwa i niezawodności w bazach danych**

Bezpieczeństwo w systemach bazodanowych obejmuje:

* **Autentykację i autoryzację:** Określenie, kto ma dostęp do danych i jaką rolę pełni.
* **Szyfrowanie:** Chroni dane przechowywane w bazie przed nieautoryzowanym dostępem.
* **Transakcje:** Zapewniają integralność danych i ich spójność (ACID).

*88. Omów pojęcie wzorca projektowego ich rodzaje oraz jego rola w procesie projektowania.*

**88. Wzorce projektowe – definicja i rola w projektowaniu**

**Wzorce projektowe** to sprawdzone, powtarzalne rozwiązania dla najczęstszych problemów projektowych w inżynierii oprogramowania. Pomagają w tworzeniu elastycznych, skalowalnych i łatwych do utrzymania systemów.

**Rodzaje wzorców:**

* **Creational (tworzeniowe):** Wzorce służące do tworzenia obiektów, np. **Factory Method**.
* **Structural (strukturalne):** Wzorce organizujące klasy i obiekty, np. **Adapter**.
* **Behavioral (behawioralne):** Wzorce definiujące komunikację między obiektami, np. **Observer**.

*89. Wyjaśnij na przykładach zasady SOLID w projektowaniu obiektowym.*

**89. Zasady SOLID w projektowaniu obiektowym**

**SOLID** to zestaw zasad ułatwiających tworzenie czytelnego i elastycznego kodu:

* **S** (Single Responsibility Principle) – Każda klasa powinna mieć tylko jedną odpowiedzialność.
* **O** (Open/Closed Principle) – Klasy powinny być otwarte na rozszerzenia, ale zamknięte na modyfikacje.
* **L** (Liskov Substitution Principle) – Obiekty klas pochodnych powinny być wymienne z obiektami klas bazowych.
* **I** (Interface Segregation Principle) – Interfejsy powinny być dopasowane do potrzeb klas.
* **D** (Dependency Inversion Principle) – Zależności powinny być skierowane na abstrakcje, a nie konkretne klasy.

*90. Co to jest język zapytań do baz danych i z jakich elementów się składa. Omów ich przeznaczenie.*

**90. Język zapytań do baz danych – składniki i przeznaczenie**

**Język zapytań do baz danych (SQL)** to język służący do komunikacji z bazami danych, umożliwiający wykonywanie operacji na danych.

**Składniki:**

* **DML (Data Manipulation Language):** Służy do manipulacji danymi (np. SELECT, INSERT, UPDATE, DELETE).
* **DDL (Data Definition Language):** Służy do definiowania struktury bazy danych (np. CREATE, ALTER, DROP).
* **DCL (Data Control Language):** Służy do kontrolowania dostępu do danych (np. GRANT, REVOKE).
* **TCL (Transaction Control Language):** Zarządza transakcjami (np. COMMIT, ROLLBACK).

**Przeznaczenie:** SQL jest używany do zarządzania bazami danych, pobierania, modyfikowania, dodawania i usuwania danych.

*91. Omów system śledzenia traksakcji w bazach danych i podaj przykłady problemów jakie on rozwiązuje.*

**91. System śledzenia transakcji w bazach danych i przykłady problemów, które rozwiązuje**

**System śledzenia transakcji** w bazach danych to mechanizm umożliwiający zarządzanie transakcjami w sposób zapewniający spójność, niezawodność i integralność danych. Główne cechy to:

* **ACID (Atomicity, Consistency, Isolation, Durability):** Zapewnia, że transakcje są wykonywane w sposób, który gwarantuje poprawność danych.
  + **Atomicity:** Transakcja jest traktowana jako całość – albo wszystkie operacje w ramach transakcji zostaną zakończone pomyślnie, albo żadna z nich nie zostanie zastosowana.
  + **Consistency:** Transakcja doprowadza system do spójnego stanu przed i po jej wykonaniu.
  + **Isolation:** Transakcje są wykonywane w izolacji od siebie, co oznacza, że wyniki jednej transakcji nie są widoczne dla innych do czasu jej zakończenia.
  + **Durability:** Po zakończeniu transakcji jej zmiany są trwale zapisane, nawet w przypadku awarii systemu.

**Problemy, które rozwiązuje:**

* **Problem braku spójności:** System śledzenia transakcji zapewnia, że dane będą spójne, nawet jeśli transakcje są przerwane przez awarię.
* **Problem nieizolowanych transakcji:** Mechanizm transakcji zapewnia, że operacje nie będą się nawzajem zakłócały.
* **Problem odzyskiwania po awarii:** Zabezpiecza dane przed utratą dzięki zapewnieniu trwałości operacji.

*92. Omów etapy inżynierii oprogramowania i wskaż jakie artefakty powstają na każdym z etapów.*

**92. Etapy inżynierii oprogramowania i artefakty powstające na każdym etapie**

**Etapy inżynierii oprogramowania:**

1. **Analiza wymagań:** Zbieranie i analizowanie wymagań systemu.
   * **Artefakty:** Dokument wymagań, analiza interesariuszy, przypadki użycia.
2. **Projektowanie systemu:** Tworzenie ogólnej architektury i szczegółowego projektu systemu.
   * **Artefakty:** Diagramy UML, dokumentacja architektury, prototypy interfejsu użytkownika.
3. **Implementacja:** Programowanie systemu zgodnie z zaprojektowaną architekturą.
   * **Artefakty:** Kod źródłowy, dokumentacja kodu, testy jednostkowe.
4. **Testowanie:** Weryfikacja i walidacja systemu pod kątem błędów i spełnienia wymagań.
   * **Artefakty:** Raporty z testów, plany testów, dokumentacja błędów.
5. **Utrzymanie:** Dbanie o system po wdrożeniu, usuwanie błędów i wdrażanie nowych funkcjonalności.
   * **Artefakty:** Zgłoszenia błędów, poprawki do kodu, dokumentacja aktualizacji.

*93. Omów techniki pozyskiwania wymagań. Wskaż, które z nich są adekwatne gdy odbiorcami rozwiązania są firmy B2B (Business-to-Business), a które gdy indywidualni użytkownicy B2C (Buisness-to-Consumer).*

**93. Techniki pozyskiwania wymagań i ich zastosowanie w B2B i B2C**

**Techniki pozyskiwania wymagań:**

1. **Wywiady:** Rozmowy z interesariuszami w celu poznania ich potrzeb i oczekiwań.
   * **B2B:** Wywiady z menedżerami, inżynierami, klientami B2B.
   * **B2C:** Wywiady z użytkownikami indywidualnymi, badania fokusowe.
2. **Warsztaty (Workshops):** Spotkania grupowe, gdzie omawia się wymagania.
   * **B2B:** Warsztaty z przedstawicielami różnych działów organizacji.
   * **B2C:** Warsztaty z użytkownikami, by lepiej zrozumieć ich potrzeby.
3. **Prototypowanie:** Tworzenie wstępnej wersji systemu, aby zobaczyć, czy spełnia oczekiwania użytkowników.
   * **B2B:** Wykorzystywane w przypadku dużych systemów B2B, np. ERP.
   * **B2C:** Prototypy aplikacji mobilnych i webowych do testowania z użytkownikami.
4. **Ankiety i kwestionariusze:** Zbieranie danych w formie pytań do szerokiego kręgu osób.
   * **B2B:** Ankiety dla przedsiębiorstw w celu zbierania informacji o preferencjach.
   * **B2C:** Kwestionariusze skierowane do użytkowników końcowych, np. aplikacji konsumenckich.

*94. Jakie funkcje powinien realizować system zarządzania bazą danych?*

**94. Funkcje systemu zarządzania bazą danych**

System zarządzania bazą danych (DBMS) pełni kluczową rolę w przechowywaniu, zarządzaniu i przetwarzaniu danych. Jego główne funkcje to:

* **Przechowywanie danych:** DBMS przechowuje dane w sposób uporządkowany, zapewniając łatwy dostęp.
* **Zarządzanie dostępem:** Kontroluje, kto ma dostęp do danych i w jakim zakresie (autentykacja, autoryzacja).
* **Manipulacja danymi:** Umożliwia dodawanie, modyfikowanie, usuwanie i wyszukiwanie danych.
* **Bezpieczeństwo:** Szyfrowanie, tworzenie kopii zapasowych, odzyskiwanie danych.
* **Zarządzanie transakcjami:** Obsługuje mechanizmy transakcyjne zapewniające integralność danych.
* **Indeksowanie:** Zoptymalizowane mechanizmy wyszukiwania danych.

*95. Podaj uwarunkowania stosowania podejścia obiektowego i strukturalnego w projektowaniu systemów informatycznych.*

**95. Uwarunkowania stosowania podejścia obiektowego i strukturalnego w projektowaniu systemów informatycznych**

**Podejście obiektowe (OOP):**

* **Uwarunkowania:** Stosowane w przypadku systemów, które wymagają dużej elastyczności, rozwoju i ponownego wykorzystania kodu. Dobre dla aplikacji złożonych, które muszą być łatwe w utrzymaniu i rozszerzaniu.
* **Przykład:** Programy oparte na aplikacjach mobilnych i rozbudowanych systemach informatycznych.

**Podejście strukturalne:**

* **Uwarunkowania:** Skierowane na proste, sekwencyjne operacje, gdzie kod jest bardziej liniowy i łatwiejszy do zrozumienia w mniejszych projektach.
* **Przykład:** Systemy wbudowane, proste aplikacje desktopowe.

*96. Jakie są kluczowe wymagania funkcjonalne i niefunkcjonalne systemu, który ma integrować różnorodne systemy informatyczne w dużej organizacji?*

**96. Kluczowe wymagania funkcjonalne i niefunkcjonalne systemu integrującego różnorodne systemy informatyczne**

**Wymagania funkcjonalne:**

* **Integracja z innymi systemami:** Możliwość wymiany danych z systemami ERP, CRM, itp.
* **Obsługa różnych formatów danych:** Konwersja i przesyłanie danych między różnymi formatami.
* **Wydajność:** Szybka wymiana danych i integracja w czasie rzeczywistym.

**Wymagania niefunkcjonalne:**

* **Bezpieczeństwo:** Silne mechanizmy szyfrowania, uwierzytelnianie użytkowników.
* **Skalowalność:** Możliwość rozbudowy systemu o dodatkowe moduły w przyszłości.
* **Wydajność:** Niski czas odpowiedzi, optymalizacja zapytań.

*97. Omów zasady działania wzorca MVC (ang. Model View Controler). W jakich projektach zastosowałbyś tę architekturę?*

**97. Zasady działania wzorca MVC (Model View Controller)**

**MVC (Model-View-Controller)** to wzorzec architektoniczny, który dzieli aplikację na trzy główne komponenty:

* **Model:** Odpowiada za logikę biznesową i dostęp do danych.
* **View:** Odpowiada za interfejs użytkownika (UI).
* **Controller:** Pośredniczy pomiędzy modelem a widokiem, zarządza logiką aplikacji.

**Zastosowanie:** MVC jest idealne w aplikacjach webowych i mobilnych, gdzie mamy wyraźny podział na dane (Model), interfejs użytkownika (View) i logikę sterującą (Controller). Przykład: aplikacje typu e-commerce, blogi.

*98. Omów główne zasady metodyk zwinnych i jakie rozwiązują problemy?*

**98. Główne zasady metodyk zwinnych i rozwiązywane problemy**

**Metodyki zwinne (Agile):**

1. **Iteracyjne i przyrostowe podejście:** Praca w krótkich iteracjach, dostarczanie działających wersji oprogramowania.
2. **Elastyczność i adaptacja:** Łatwość w adaptacji do zmieniających się wymagań.
3. **Współpraca z klientem:** Bliska współpraca z interesariuszami w celu lepszego zrozumienia ich potrzeb.
4. **Skupienie na wartości biznesowej:** Priorytetem jest dostarczenie wartościowych funkcji.

**Rozwiązywane problemy:**

* Szybka reakcja na zmiany w wymaganiach.
* Zwiększenie transparentności procesu tworzenia oprogramowania.
* Ułatwienie dostosowywania oprogramowania do rzeczywistych potrzeb użytkowników.

*99. Z jakimi obszarami związane są wymagania niefunkcjonale i jakich? Podaj przykłady poprawnie sformułowanych wymagań.*

**99. Wymagania niefunkcjonalne i przykłady**

**Wymagania niefunkcjonalne** odnoszą się do jakości systemu, takich jak:

* **Wydajność:** Przykład: System powinien obsługiwać 10 000 zapytań na minutę.
* **Skalowalność:** Przykład: System powinien być w stanie obsłużyć 1 milion użytkowników.
* **Bezpieczeństwo:** Przykład: Wszystkie dane powinny być szyfrowane za pomocą algorytmu AES-256.

*100. Wymień i opisz cechy różnych typów aplikacji mobilnych (natywne, hybrydowe, wieloplatformowe itp.).*

**100. Cechy różnych typów aplikacji mobilnych**

1. **Aplikacje natywne:**
   * Działają na określonej platformie (iOS, Android).
   * Wysoka wydajność i dostęp do pełnej funkcjonalności urządzenia.
   * Wymagają osobnych wersji dla każdej platformy.
2. **Aplikacje hybrydowe:**
   * Kombinacja aplikacji natywnych i webowych.
   * Jedna baza kodu dla wielu platform.
   * Mniejsza wydajność w porównaniu z aplikacjami natywnymi.
3. **Aplikacje wieloplatformowe:**
   * Pisane raz i uruchamiane na wielu platformach (np. Xamarin, React Native).
   * Oferują kompromis między wydajnością a oszczędnością czasu rozwoju.

**Zagadnienia dla kierunku Informatyka,**

**specjalność: Aplikacje mobilne i bazy danych**

*1. Jakie są metody zarządzania pamięcią w aplikacjach mobilnych?*

**1. Metody zarządzania pamięcią w aplikacjach mobilnych**

W aplikacjach mobilnych zarządzanie pamięcią jest kluczowym elementem optymalizacji, ponieważ urządzenia mobilne mają ograniczone zasoby. Główne metody zarządzania pamięcią to:

* **Garbage Collection (GC):** Systemy operacyjne, takie jak Android i iOS, automatycznie zarządzają pamięcią za pomocą GC, który usuwa nieużywane obiekty z pamięci. Programista nie musi manualnie zwalniać pamięci, ale musi zadbać o unikanie wycieków pamięci.
* **Zarządzanie zasobami:** Programiści muszą dbać o poprawne zwalnianie zasobów, takich jak obrazy, pliki wideo czy połączenia sieciowe, aby uniknąć wycieków pamięci.
* **Zarządzanie pamięcią podręczną:** Używanie pamięci podręcznej do przechowywania danych, takich jak obrazy czy dane z sieci, aby zmniejszyć liczbę operacji I/O i zaoszczędzić pamięć.
* **Ograniczanie liczby obiektów:** Optymalizacja alokacji pamięci poprzez minimalizowanie liczby tworzonych obiektów i dbanie o ich ponowne użycie.

*2. W jaki sposób można zoptymalizować wydajność aplikacji iOS, proszę przedstawić różne techniki i narzędzia używane do monitorowania i poprawy wydajności aplikacji, takie jak profilowanie, zarządzanie pamięcią i optymalizacja renderowania interfejsu użytkownika?*

**2. Optymalizacja wydajności aplikacji iOS**

Aby zoptymalizować wydajność aplikacji na iOS, programiści używają różnych technik i narzędzi:

* **Profilowanie:** Xcode oferuje narzędzia takie jak Instruments, które umożliwiają profilowanie aplikacji pod kątem zużycia CPU, pamięci i innych zasobów. Profilowanie pomaga zidentyfikować miejsca w aplikacji, które wymagają optymalizacji.
* **Zarządzanie pamięcią:** iOS zarządza pamięcią automatycznie, ale programiści muszą unikać wycieków pamięci i odpowiednio zarządzać zasobami. Używanie ARC (Automatic Reference Counting) w iOS pozwala na automatyczne zwalnianie pamięci, ale należy pamiętać o odpowiednim zarządzaniu cyklem życia obiektów.
* **Optymalizacja renderowania UI:** iOS oferuje różne narzędzia do analizy renderowania interfejsu użytkownika, takie jak Instruments (Core Animation) do monitorowania opóźnień renderowania UI. Optymalizacja animacji, redukcja liczby widoków w hierarchii UI i używanie lazy loading może znacznie poprawić wydajność.
* **Redukcja operacji synchronizacji:** Zbyt częsta synchronizacja z serwerem może prowadzić do spowolnienia aplikacji. Optymalizowanie czasów synchronizacji i zarządzanie nimi, np. w tle, może poprawić wydajność.

*3. Jakie techniki i narzędzia można zastosować, aby zoptymalizować wydajność aplikacji mobilnej, minimalizując czas ładowania i zużycie zasobów?*

**3. Optymalizacja wydajności aplikacji mobilnej – minimalizacja czasu ładowania i zużycia zasobów**

Aby zoptymalizować wydajność aplikacji mobilnej, można zastosować:

* **Lazy loading:** Wczytywanie danych lub widoków tylko wtedy, gdy są one potrzebne, zamiast ładowania wszystkiego na raz.
* **Kompresja zasobów:** Kompresowanie obrazów, plików wideo i innych zasobów w celu zmniejszenia ich rozmiaru, co pozwala na szybsze ładowanie aplikacji.
* **Optymalizacja zapytań sieciowych:** Redukcja liczby zapytań sieciowych, używanie odpowiednich algorytmów kompresji, minimalizacja danych przesyłanych między urządzeniem a serwerem.
* **Użycie pamięci podręcznej:** Przechowywanie często używanych danych (np. obrazy, dane z API) w pamięci podręcznej, co przyspiesza dostęp do nich i zmniejsza obciążenie sieci.

*4. Jak skutecznie zarządzać stanem aplikacji mobilnej, aby zapewnić płynne działanie i zachowanie danych podczas zmiany konfiguracji, takich jak obrót ekranu?*

**4. Zarządzanie stanem aplikacji mobilnej – zmiany konfiguracji**

Aby zapewnić płynne działanie aplikacji i zachowanie danych podczas zmiany konfiguracji (np. obrót ekranu), można zastosować:

* **Przechowywanie stanu w pamięci:** Korzystanie z mechanizmów przechowywania stanu (np. onSaveInstanceState w Androidzie lub state restoration w iOS), które umożliwiają zachowanie danych i ponowne załadowanie ich po zmianie konfiguracji.
* **Zarządzanie danymi w bazach danych:** Przechowywanie kluczowych danych aplikacji w bazach danych lub innych trwałych magazynach, aby były dostępne po ponownym uruchomieniu aplikacji.
* **Unikanie blokowania interfejsu:** Aplikacja powinna wykorzystywać asynchroniczne operacje w celu uniknięcia blokowania interfejsu użytkownika podczas zmian konfiguracji.

*5. Jakie są najlepsze praktyki integracji aplikacji mobilnej z backendem, aby zapewnić niezawodną komunikację i synchronizację danych?*

**5. Najlepsze praktyki integracji aplikacji mobilnej z backendem**

Aby zapewnić niezawodną komunikację i synchronizację danych, należy:

* **Używać bezpiecznych połączeń (HTTPS):** Zapewnienie bezpieczeństwa danych przesyłanych między aplikacją mobilną a backendem.
* **Wykorzystanie technologii push (np. FCM, APNs):** Umożliwia to synchronizację danych w czasie rzeczywistym.
* **Zoptymalizowane API:** Tworzenie RESTful API, które jest szybkie, wydajne i łatwe w integracji z aplikacjami mobilnymi.
* **Używanie lokalnych baz danych:** Zapewnia to możliwość przechowywania danych offline i synchronizacji ich, gdy połączenie z internetem będzie dostępne.

*6. Jakie techniki można zastosować, aby minimalizować zużycie baterii przez aplikację mobilną, jednocześnie zapewniając pełną funkcjonalność?*

**6. Minimalizacja zużycia baterii przez aplikację mobilną**

Aby minimalizować zużycie baterii, należy:

* **Optymalizować operacje w tle:** Ograniczenie częstotliwości wykonywania operacji w tle (np. zapytań sieciowych).
* **Wykorzystywanie lokalizacji tylko wtedy, gdy jest to konieczne:** Używanie GPS tylko w razie potrzeby i w trybie oszczędzania energii.
* **Zarządzanie powiadomieniami:** Ograniczenie liczby powiadomień, które mogą wpływać na zużycie energii.
* **Używanie trybu oszczędzania energii:** Wiele urządzeń mobilnych ma wbudowane tryby oszczędzania energii, które ograniczają działanie aplikacji w tle.

*7. Jak zarządzać zasobami aplikacji mobilnej, takimi jak obrazy, dźwięki i pliki wideo, aby zoptymalizować jej wydajność i rozmiar?*

**7. Zarządzanie zasobami aplikacji mobilnej (obrazy, dźwięki, wideo)**

Aby zoptymalizować wydajność i rozmiar aplikacji:

* **Kompresja zasobów:** Kompresowanie obrazów (np. format WebP), dźwięków i plików wideo, aby zmniejszyć ich rozmiar bez utraty jakości.
* **Używanie odpowiednich rozdzielczości:** Ładowanie obrazów o odpowiednich rozdzielczościach, które są dostosowane do ekranu urządzenia.
* **Lazy loading zasobów:** Ładowanie zasobów tylko wtedy, gdy są one wymagane, zamiast ładowania ich na starcie aplikacji.

*8. Jak zaprojektować aplikację mobilną, która działa efektywnie zarówno w trybie online, jak i offline, zapewniając synchronizację danych, gdy połączenie jest dostępne?*

**8. Projektowanie aplikacji mobilnej do pracy online i offline**

Aby aplikacja działała efektywnie zarówno w trybie online, jak i offline:

* **Lokalne przechowywanie danych:** Używanie baz danych lokalnych (np. SQLite, CoreData), które przechowują dane offline.
* **Synchronizacja danych:** Zapewnienie mechanizmów synchronizacji, które umożliwiają przesyłanie danych na serwer, gdy połączenie jest dostępne.
* **Zarządzanie stanem offline:** Informowanie użytkownika o braku połączenia i umożliwienie kontynuowania pracy w trybie offline.

*9. Jakie są skuteczne modele monetyzacji aplikacji mobilnych, takie jak reklamy, zakupy w aplikacji, subskrypcje, i jakie są ich zalety i wady w kontekście doświadczenia użytkownika?*

**9. Modele monetyzacji aplikacji mobilnych**

Modele monetyzacji:

* **Reklamy:** Wstawianie reklam w aplikacji (np. Google AdMob). Zaletą jest brak konieczności płacenia przez użytkownika, wadą – możliwość irytowania użytkowników.
* **Zakupy w aplikacji:** Umożliwienie użytkownikom kupowania treści lub funkcji w aplikacji. Zaletą jest łatwy dostęp do dodatkowych funkcji, ale wymaga dokładnego planowania modeli cenowych.
* **Subskrypcje:** Oferowanie subskrypcji na dostęp do treści lub funkcji w aplikacji. Zaletą jest przewidywalny dochód, ale może odstraszyć użytkowników, którzy nie są gotowi na regularne płatności.

*10. Jak projektować i implementować funkcje współdzielenia zawartości w aplikacjach mobilnych, takie jak udostępnianie zdjęć, plików i treści w mediach społecznościowych?*

**10. Funkcje współdzielenia zawartości w aplikacjach mobilnych**

Aby zaimplementować funkcje współdzielenia zawartości:

* **Integracja z mediami społecznościowymi:** Użycie API takich jak Facebook SDK, Twitter SDK do łatwego udostępniania treści.
* **Udostępnianie plików:** Możliwość wysyłania plików za pomocą systemów udostępniania plików, takich jak AirDrop (iOS) lub Intent (Android).
* **Tworzenie uniwersalnych przycisków udostępniania:** Umożliwienie użytkownikom udostępniania treści w różnych serwisach społecznościowych bez konieczności przełączania aplikacji.

*11. Jakie są wyzwania związane z optymalizacją aplikacji mobilnych pod kątem różnych wersji systemów operacyjnych i jak z nimi sobie radzić?*

**11. Wyzwania związane z optymalizacją aplikacji mobilnych pod kątem różnych wersji systemów operacyjnych i jak z nimi sobie radzić?**

Wyzwania:

* **Fragmentacja urządzeń:** Różne wersje systemów operacyjnych, urządzenia o różnych rozdzielczościach ekranów, procesorach, pojemności pamięci itp. mogą prowadzić do problemów z kompatybilnością aplikacji.
* **Różnice w interfejsie użytkownika:** Systemy operacyjne mogą mieć różne wytyczne dotyczące interfejsu użytkownika, co wymaga dostosowania aplikacji do specyficznych wymagań platformy.

Jak radzić sobie z wyzwaniami:

* **Testowanie na różnych urządzeniach:** Regularne testowanie aplikacji na różnych wersjach systemów operacyjnych i urządzeniach za pomocą emulatorów oraz rzeczywistych urządzeń.
* **Używanie narzędzi takich jak Firebase czy TestFlight (iOS) oraz Android Emulator:** Pomagają w testowaniu aplikacji na różnych wersjach systemów operacyjnych.
* **Responsywny design:** Użycie rozwiązań responsywnych, które umożliwiają dostosowanie UI do różnych rozdzielczości ekranów.
* **Wersjonowanie kodu:** Korzystanie z narzędzi do wersjonowania kodu oraz wtyczek umożliwiających wsparcie dla starszych wersji systemów operacyjnych.

*12. Jak projektować aplikacje mobilne z myślą o skalowalności, aby rosnąca liczba użytkowników nie wpływała negatywnie na wydajność i dostępność aplikacji?*

**12. Projektowanie aplikacji mobilnych z myślą o skalowalności**

Aby zaprojektować aplikację mobilną, która będzie skalowalna:

* **Modularność:** Tworzenie aplikacji w sposób modularny, dzieląc ją na mniejsze komponenty. Umożliwia to łatwiejsze zarządzanie aplikacją przy wzroście liczby użytkowników.
* **Chmurowa infrastruktura:** Wykorzystanie chmurowych usług backendowych (np. AWS, Google Cloud) do przechowywania danych i obsługi logiki biznesowej, co zapewnia skalowalność.
* **Optymalizacja zapytań sieciowych:** Używanie algorytmów kompresji oraz zmniejszenie liczby zapytań do serwera w celu zminimalizowania opóźnień i zmniejszenia obciążenia serwerów.
* **Zastosowanie CDN:** Korzystanie z sieci dostarczania treści (CDN), aby przyspieszyć ładowanie zasobów aplikacji z różnych lokalizacji geograficznych.
* **Użycie load balancerów i rozproszonych baz danych:** Rozdzielanie obciążenia między serwery i stosowanie baz danych, które obsługują dużą ilość danych oraz zapewniają redundancję.

*13. Czym jest Android Studio i jakie są jego główne funkcje?*

**13. Android Studio i jego główne funkcje**

**Android Studio** to oficjalne środowisko programistyczne dla systemu Android. Jego główne funkcje to:

* **Edytor kodu:** Zaawansowany edytor z funkcjami automatycznego uzupełniania kodu, refaktoryzacji oraz diagnostyki błędów.
* **Emulator Android:** Umożliwia uruchomienie aplikacji na różnych wersjach Androida i urządzeniach wirtualnych.
* **Narzędzia do debugowania:** Android Studio zapewnia narzędzia do debugowania aplikacji, które pomagają śledzić błędy i analizować logi.
* **Wbudowane narzędzia do testowania:** Integracja z narzędziami do testowania, zarówno jednostkowymi, jak i UI, umożliwia sprawdzanie jakości aplikacji.
* **Designer UI:** Graficzny edytor do tworzenia interfejsów użytkownika w technologii XML oraz Jetpack Compose.

*14. Jakie są kluczowe różnice między frameworkiem UIKit a SwiftUI w kontekście tworzenia interfejsów użytkownika w aplikacjach iOS, proszę omówić, jakie korzyści i wyzwania wiążą się z korzystaniem z każdego z tych frameworków oraz w jakich scenariuszach jeden może być preferowany nad drugim?*

**14. Różnice między UIKit a SwiftUI w kontekście tworzenia interfejsów użytkownika**

**UIKit**:

* Tradycyjny framework UI używany w iOS.
* Wymaga ręcznego zarządzania hierarchią widoków i kontrolerów.
* Bardziej rozbudowany i elastyczny, ale ma wyższą krzywą uczenia się.

**SwiftUI**:

* Nowoczesny framework oparty na deklaratywnym podejściu do tworzenia UI.
* Umożliwia łatwiejsze tworzenie UI poprzez deklarację stanu aplikacji i jego aktualizację.
* Lepsza integracja z nowymi funkcjami systemu iOS, takich jak Dark Mode i Accessibility.

**Korzyści i wyzwania**:

* **UIKit**: Jest bardziej dojrzały, szeroko wspierany i elastyczny, ale wymaga więcej kodu do zarządzania widokami.
* **SwiftUI**: Uproszczony kod, lepsze wsparcie dla dynamicznych interfejsów, ale może mieć ograniczenia w starszych wersjach iOS (przynajmniej na początku).

**Preferencje**: SwiftUI może być preferowany w nowych projektach, natomiast UIKit pozostaje preferowaną opcją w starszych aplikacjach, gdzie konieczne jest utrzymanie kompatybilności wstecznej.

*15. Omów rolę bibliotek wsparcia wzornictwa w projektowaniu i programowaniu aplikacji mobilnych dla systemu Android.*

**15. Rola bibliotek wsparcia wzornictwa w projektowaniu aplikacji Android**

Biblioteki wsparcia wzornictwa w Androidzie oferują narzędzia i komponenty ułatwiające projektowanie interfejsów zgodnych z wytycznymi Material Design. Do najważniejszych funkcji należą:

* **Zestaw komponentów UI:** Takich jak widżety przycisków, przełączników, pasków narzędzi, które pozwalają na spójne projektowanie aplikacji.
* **Material Theming:** Umożliwia łatwe dostosowanie wyglądu aplikacji poprzez palety kolorów, style typografii i kształty elementów UI.
* **Gesty i animacje:** Ułatwiają dodawanie animacji, jak przejścia między ekranami czy efekty dotykowe, które poprawiają interaktywność aplikacji.
* **Zgodność z wersjami Androida:** Zapewniają wsparcie dla różnych wersji systemu Android, co pozwala na tworzenie spójnego UI, niezależnie od wersji OS.

*16. Jakie są główne rodzaje testów w testowaniu aplikacji mobilnych?*

**16. Rodzaje testów w testowaniu aplikacji mobilnych**

* **Testy jednostkowe (Unit tests):** Testowanie poszczególnych funkcji i metod aplikacji w celu sprawdzenia, czy działają one zgodnie z oczekiwaniami.
* **Testy integracyjne (Integration tests):** Testowanie interakcji pomiędzy różnymi modułami lub komponentami aplikacji.
* **Testy funkcjonalne (Functional tests):** Sprawdzanie, czy aplikacja działa zgodnie z wymaganiami i specyfikacjami.
* **Testy UI (User Interface tests):** Testowanie interakcji użytkownika z aplikacją, sprawdzanie, czy elementy UI są odpowiednio rozmieszczone i czy są intuicyjne.
* **Testy wydajnościowe (Performance tests):** Mierzenie szybkości działania aplikacji, jej odpowiedzi na różne operacje, np. ładowanie danych.
* **Testy bezpieczeństwa (Security tests):** Sprawdzanie, czy aplikacja jest odporna na ataki i zapewnia bezpieczeństwo danych użytkownika.

*17. Jakie są najlepsze praktyki w zakresie testowania aplikacji iOS, zwłaszcza w kontekście testów jednostkowych i testów UI*

**17. Najlepsze praktyki w zakresie testowania aplikacji iOS**

* **Testy jednostkowe:** Używanie frameworków takich jak XCTest do testowania poszczególnych funkcji i metod aplikacji.
* **Testy UI:** Automatyzacja testów UI za pomocą narzędzi takich jak XCUITest do interakcji z interfejsem aplikacji.
* **Mockowanie danych:** Używanie mocków i stubów do symulacji danych, które aplikacja przetwarza.
* **Continuous Integration (CI):** Używanie narzędzi CI, takich jak Jenkins lub Bitrise, do automatyzacji testów podczas każdego commita w kodzie.
* **Testy wydajnościowe:** Używanie narzędzi takich jak Instruments w Xcode, które pozwala na śledzenie użycia CPU, pamięci, animacji i renderowania UI w aplikacji.

*18. Jakie podejścia i narzędzia można wykorzystać do skutecznego testowania aplikacji mobilnych, obejmującego testy jednostkowe, integracyjne oraz testy UI?*

**18. Testowanie aplikacji mobilnych – narzędzia i podejścia**

* **Testy jednostkowe:** Narzędzia takie jak JUnit (Android) lub XCTest (iOS) do testowania pojedynczych funkcji.
* **Testy integracyjne:** Użycie testów integracyjnych do sprawdzania poprawności działania aplikacji z backendem oraz integracji z bazami danych.
* **Testy UI:** Narzędzia jak Espresso (Android) i XCUITest (iOS) umożliwiają automatyczne testowanie interakcji użytkownika z aplikacją.
* **Symulowanie różnych scenariuszy:** Korzystanie z emulatorów i rzeczywistych urządzeń, aby testować aplikację w różnych warunkach (np. w trybie offline).

*19. Jak zaprojektować aplikację mobilną, aby była dostępna dla użytkowników z różnymi niepełnosprawnościami, zgodnie z wytycznymi WCAG (Web Content Accessibility Guidelines)?*

**19. Dostępność aplikacji mobilnej – wytyczne WCAG**

* **Wytyczne WCAG (Web Content Accessibility Guidelines):** Pomagają w tworzeniu aplikacji dostępnych dla osób z niepełnosprawnościami.
* **Użycie odpowiednich kontrastów:** Zapewnienie odpowiednich kontrastów kolorów w interfejsie.
* **Wspieranie narracji głosowej:** Wykorzystanie VoiceOver (iOS) i TalkBack (Android) dla osób niewidomych.
* **Ułatwienia w nawigacji:** Umożliwienie nawigacji za pomocą klawiatury lub innych urządzeń wspomagających.
* **Skalowanie tekstów:** Zapewnienie opcji powiększania tekstu w aplikacji.

*20. Jak zaprojektować intuicyjny i responsywny interfejs użytkownika, który zapewni pozytywne doświadczenia użytkownika na różnych urządzeniach i rozmiarach ekranów?*

**20. Projektowanie intuicyjnego i responsywnego UI**

* **Responsywny design:** Użycie technik takich jak Auto Layout (iOS) czy ConstraintLayout (Android) do dostosowania układu do różnych rozmiarów ekranów.
* **Testowanie na różnych urządzeniach:** Regularne testowanie aplikacji na różnych urządzeniach o różnych rozdzielczościach ekranów.
* **Minimalizm w designie:** Prosty, przejrzysty design z jasnymi i łatwymi do zrozumienia interakcjami.
* **Interaktywne elementy:** Używanie animacji i przejść w sposób, który poprawia użyteczność i wrażenia użytkownika.

*21. Jak zapewnić, aby aplikacja mobilna była zgodna z trendami UX/UI, takimi jak dark mode, gesty nawigacyjne czy minimalistyczny design?*

**21. Jak zapewnić, aby aplikacja mobilna była zgodna z trendami UX/UI, takimi jak dark mode, gesty nawigacyjne czy minimalistyczny design?**

Aby aplikacja była zgodna z nowoczesnymi trendami UX/UI:

* **Dark Mode:** Implementacja trybu ciemnego zgodnie z wytycznymi systemów operacyjnych (np. iOS i Android). Użycie systemowych trybów zmiany motywów (np. UIUserInterfaceStyle w iOS) pozwala na łatwą implementację, która będzie zmieniać wygląd aplikacji w zależności od preferencji użytkownika.
* **Gesty nawigacyjne:** Integracja gestów takich jak przewijanie, przesuwanie, czy gesty dotykowe (np. Swipe, Tap, Pinch) do nawigacji, które poprawiają interakcję użytkownika. Należy również zadbać o ich spójność z systemem operacyjnym, aby uniknąć konfliktów z wbudowanymi gestami.
* **Minimalistyczny design:** Redukowanie elementów graficznych do niezbędnego minimum, co poprawia czytelność i nawigację. Zastosowanie dużych czcionek, prostych ikon oraz eliminacja zbędnych animacji zwiększa efektywność UX.

*22. Jak zapewnić, aby aplikacja mobilna była przyjazna dla użytkowników starszych, uwzględniając ich specyficzne potrzeby i preferencje dotyczące interfejsu użytkownika?*

**22. Jak zapewnić, aby aplikacja mobilna była przyjazna dla użytkowników starszych, uwzględniając ich specyficzne potrzeby i preferencje dotyczące interfejsu użytkownika?**

Aby aplikacja była przyjazna dla starszych użytkowników:

* **Duża czcionka i kontrast:** Zapewnienie dużych i czytelnych czcionek oraz wysokiego kontrastu, by poprawić czytelność.
* **Prosty, intuicyjny interfejs:** Stosowanie prostych i zrozumiałych interakcji, unikanie nadmiernej liczby opcji na ekranie. Pomocne może być użycie jednego ekranu głównego z wyraźnymi przyciskami do podstawowych funkcji.
* **Zwiększona responsywność dotyku:** Ułatwienie obsługi aplikacji osobom z ograniczoną sprawnością motoryczną poprzez większe przyciski i obszary dotykowe.
* **Integracja z funkcjami systemowymi:** Wykorzystanie funkcji systemowych takich jak Zoom, VoiceOver (iOS) czy TalkBack (Android) do poprawy dostępności aplikacji.
* **Instrukcje i wsparcie:** Proste, zrozumiałe komunikaty i dostępność samouczków lub wsparcia na każdym etapie korzystania z aplikacji.

*23. Jakie są kluczowe zasady projektowania interfejsu użytkownika (UI) i doświadczeń użytkownika (UX) w kontekście aplikacji iOS, proszę omówić, jak zasady Human Interface Guidelines (HIG) Apple wpływają na projektowanie aplikacji oraz jak można je zastosować, aby tworzyć intuicyjne i atrakcyjne interfejsy użytkownika*

**23. Kluczowe zasady projektowania UI/UX w aplikacjach iOS i wpływ Human Interface Guidelines (HIG) Apple na projektowanie**

**Zasady Human Interface Guidelines (HIG):**

* **Spójność:** Zgodność z systemowym wyglądem, zapewnienie użytkownikowi intuicyjnego i spójnego doświadczenia z innymi aplikacjami i systemem.
* **Zrozumiałość:** Aplikacja powinna mieć logiczną strukturę, nawigację, a przyciski i elementy sterujące powinny być łatwe do znalezienia i używania.
* **Czytelność:** Użycie odpowiedniego rozmiaru czcionek, kontrastu oraz przestrzeni pomiędzy elementami interfejsu.
* **Feedback:** Aplikacja powinna reagować na akcje użytkownika, np. przez animacje, wibracje lub dźwięki, co zwiększa komfort użytkowania.
* **Dostosowanie do urządzenia:** UI powinno dostosować się do różnych rozdzielczości ekranów, formatu urządzenia (iPhone, iPad), oraz wykorzystać natywne gesty.
* **Łatwość używania:** Ułatwienia takie jak prostota w obsłudze aplikacji, możliwość szybkiego dostępu do głównych funkcji oraz ograniczenie złożoności interfejsu.

*24. Jakie są kluczowe wyzwania i potencjalne problemy związane z wydajnością oraz poprawnością danych przy stosowaniu funkcji agregujących w złożonych zapytaniach bazodanowych?*

**24. Wyzwania związane z wydajnością oraz poprawnością danych przy funkcjach agregujących w złożonych zapytaniach bazodanowych**

**Wyzwania:**

* **Wydajność:** Agregowanie dużych zbiorów danych może prowadzić do dużego obciążenia serwera, zwłaszcza w przypadku operacji takich jak JOIN, GROUP BY, HAVING przy dużych tabelach.
* **Poprawność danych:** Złożone zapytania mogą prowadzić do błędnych wyników, jeśli dane nie są poprawnie zgrupowane lub zapytania są źle zoptymalizowane.
* **Optymalizacja:** Zapytania wymagające agregacji mogą spowalniać aplikację. Rozwiązaniem jest stosowanie indeksów, ograniczanie liczby agregacji na dużych tabelach oraz wykorzystywanie zapytań z limitami.

**Zalecane rozwiązania:**

* **Indeksy:** Stosowanie indeksów na kolumnach, które są używane w zapytaniach agregujących.
* **Optymalizacja zapytań:** Zoptymalizowanie zapytań, np. przez użycie LIMIT lub dzielenie zapytań na mniejsze operacje, aby zmniejszyć obciążenie serwera.
* **Cache'owanie wyników:** Wykorzystanie cache'owania wyników dla zapytań agregujących, zwłaszcza dla danych, które rzadko się zmieniają.

*25. Główne cechy baz danych typu NoSQL ze szczególnym uwzględnieniem założeń dotyczących ich rozproszenia*

**25. Główne cechy baz danych typu NoSQL ze szczególnym uwzględnieniem założeń dotyczących ich rozproszenia**

**Cechy baz NoSQL:**

* **Elastyczność schematów:** Brak sztywno zdefiniowanego schematu bazy danych pozwala na przechowywanie różnorodnych typów danych (np. JSON, BSON).
* **Wysoka dostępność:** NoSQL zapewnia rozproszone przechowywanie danych na wielu węzłach, co poprawia dostępność i odporność na awarie.
* **Skalowalność:** Umożliwia łatwe poziome skalowanie poprzez dodawanie nowych serwerów (sharding), co pozwala obsługiwać duże ilości danych.
* **Replikacja:** Dane mogą być replikowane na różnych serwerach, co zapewnia ich bezpieczeństwo i dostępność.

**Założenia rozproszenia:**

* **Partycjonowanie danych (Sharding):** Dzieli dane na fragmenty (shardy), które są przechowywane na różnych serwerach, aby zwiększyć wydajność.
* **Replikacja:** NoSQL często stosuje replikację danych na różnych węzłach, co zapewnia odporność na awarie i zwiększa dostępność.

*26. Mechanizmy zapewniania bezpieczeństwa i niezawodności w systemach zarządzania bazami danych*

**26. Mechanizmy zapewniania bezpieczeństwa i niezawodności w systemach zarządzania bazami danych**

**Mechanizmy zapewniające bezpieczeństwo:**

* **Autoryzacja i uwierzytelnianie:** Stosowanie systemów autoryzacji (np. Role-Based Access Control - RBAC) i szyfrowania danych.
* **Szyfrowanie:** Używanie szyfrowania danych w spoczynku (np. AES) oraz szyfrowanie komunikacji z bazą danych (SSL/TLS).
* **Audit log:** Rejestrowanie operacji na bazie danych w celu monitorowania i wykrywania podejrzanych działań.
* **Kontrola dostępu:** Stosowanie uprawnień na poziomie tabel, rekordów oraz operacji (np. SELECT, INSERT).

**Mechanizmy niezawodności:**

* **Transakcje ACID:** Zapewnienie integralności danych za pomocą transakcji, które gwarantują atomowość, spójność, izolację i trwałość.
* **Replikacja:** Duplikowanie danych na różnych serwerach w celu zapewnienia dostępności w przypadku awarii.
* **Backup i odzyskiwanie:** Regularne wykonywanie kopii zapasowych bazy danych oraz testowanie procesu odzyskiwania danych.

*27. Główne cechy baz danych typu NoSQL ze szczególnym uwzględnieniem założeń dotyczących ich rozproszenia*

**27. Główne cechy baz danych typu NoSQL – rozproszenie**

(Szczegóły powtórzone w punkcie 25)

*28. Rola i przykłady silników do odwzorowań obiektowo-relacyjnych (ORM)*

**28. Rola i przykłady silników ORM (Object-Relational Mapping)**

Silniki ORM umożliwiają mapowanie obiektów w programowaniu obiektowym na tabele w relacyjnej bazie danych, co upraszcza pracę z bazą danych w aplikacjach. Przykłady ORM:

* **Hibernate (Java):** Umożliwia mapowanie klas Java do tabel w bazach SQL, automatyzując operacje CRUD.
* **Entity Framework (C#):** Narzędzie ORM dla platformy .NET, pozwala na manipulację danymi w bazie danych przy użyciu obiektów C#.
* **Django ORM (Python):** Zintegrowane w Django ORM, umożliwia interakcję z bazą danych SQL za pomocą Pythonowych obiektów.

*29. Sposoby zapewniania wydajności i strojenia w systemach zarządzania bazami danych*

**29. Sposoby zapewniania wydajności i strojenia w systemach baz danych**

* **Indeksowanie:** Tworzenie odpowiednich indeksów na kolumnach, które są używane w zapytaniach.
* **Optymalizacja zapytań:** Używanie EXPLAIN w SQL do analizy zapytań i identyfikowania nieoptymalnych operacji.
* **Cache'owanie wyników:** Przechowywanie często używanych wyników zapytań w pamięci podręcznej.
* **Sharding i replikacja:** Dzielnie danych na różne serwery w celu zwiększenia dostępności i wydajności.

*30. Jakie są kluczowe strategie i techniki zabezpieczania aplikacji mobilnych przed zagrożeniami, takimi jak ataki typu man-in-the-middle, kradzież danych czy malware?*

**30. Techniki zabezpieczania aplikacji mobilnych przed zagrożeniami**

* **Szyfrowanie danych:** Użycie szyfrowania zarówno na poziomie aplikacji (AES), jak i podczas przesyłania danych (SSL/TLS).
* **Ochrona przed atakami Man-in-the-Middle:** Użycie certyfikatów SSL/TLS do zabezpieczenia komunikacji między aplikacją a serwerem.
* **Wykrywanie malware:** Implementacja narzędzi do wykrywania złośliwego oprogramowania, jak również technik antypirackich.
* **Bezpieczne przechowywanie danych:** Unikanie przechowywania wrażliwych informacji w pamięci urządzenia, zamiast tego używanie bezpiecznych magazynów danych, takich jak Keychain (iOS) lub Keystore (Android).