Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εαρινό Εξάμηνο 2023-2024

ΤΕΧΝΟΛΟΓΙΕΣ ΥΠΗΡΕΣΙΩΝ ΛΟΓΙΣΜΙΚΟΥ

Λύσεις Θεμάτων

Ιωάννης (Χουάν) Τσαντήλας
03120883

Contents

Κανονική 21

Ερώτημα 4

Η χρήση orchestrators είναι επιλογή που μπορεί να διασφαλίσει καλύτερη διαθεσιμότητα της εφαρμογής σε περίπτωση αποτυχίας κόμβων σε μια αρχιτεκτονική microservices, απ' ότι η χρήση choreographer.

1. Σωστό.
2. Λάθος.

Λύση

Σωστό. Η χρήση orchestrators μπορεί να εξασφαλίσει καλύτερη διαθεσιμότητα της εφαρμογής σε περίπτωση αποτυχίας κόμβου, επειδή οι orchestrators διαχειρίζονται και παρακολουθούν κεντρικά τις υπηρεσίες, χειριζόμενοι αυτόματα τις αποτυχίες και το scaling. Οι choreographers, από την άλλη πλευρά, βασίζονται σε decentralized coordination, ο οποίος μπορεί να μην ανταποκρίνεται τόσο αποτελεσματικά σε αποτυχίες κόμβων.

Ερώτημα 5

Η χρήση διακριτών διεπαφών (interfaces) όπως π.χ. REST:

1. Καμία από τις υπόλοιπες απαντήσεις δεν είναι σωστή.
2. Διευκολύνει την επαναχρησιμοποίηση τμημάτων λογισμικού με αντίβαρο τη μη βέλτιστη χρήση πόρων.
3. Μπορεί να γίνει μόνο για συγκεκριμένα πρότυπα σχεδίασης (design patterns).
4. Διευκολύνει την υλοποίηση κληρονομικότητας κατ' αντιστοιχία με το αντικειμενοστρεφές παράδειγμα προγραμματισμού.

Λύση

Το **(2)**. Η χρήση διακριτών διεπαφών, όπως η REST, διευκολύνει την επαναχρησιμοποίηση στοιχείων λογισμικού με το penalty της μη βέλτιστης χρήσης πόρων. Τα RESTful APIs επιτρέπουν modularity και reusability, αλλά μπορεί να συνεπάγονται πρόσθετη καθυστέρηση δικτύου και επιπλέον χρήση πόρων σε σύγκριση με τις πιο στενά συνδεδεμένες αλληλεπιδράσεις.

Ερώτημα 6

Ποιο από τα παρακάτω επιθυμητά χαρακτηριστικά ενός καλού test έχει τη ΜΙΚΡΟΤΕΡΗ αξία;

1. Να ολοκληρώνεται γρήγορα.
2. Να μην εξαρτάται η καλή λειτουργία του από άλλα τεστ.
3. Να μπορεί να ελεγχθεί μηχανικά η επιτυχία του.
4. Να μην ελέγχει κάτι που έχει ήδη ελεγχθεί από άλλα τεστ.

Λύση

Το **(1)**.

Ερώτημα 7

Η αρχιτεκτονική SOA συνδέει μέσω ενός Service Bus εφαρμογές που:

1. Κάθε μία μπορεί να υλοποιείται με οποιαδήποτε τεχνολογία, αρκεί να συνδέεται στο Service Bus σύμφωνα με τις απαιτήσεις τής SOA.
2. Κάθε μία πρέπει να αποτελείται από "μεγάλα" συστατικά (components, "μεγάλα" με την έννοια του πλήθους των λειτουργιών) και όλες μοιράζονται ένα κοινό επίπεδο διαχείρισης δεδομένων.
3. Κάθε μία μπορεί να υλοποιείται μόνο με τις τεχνολογίες που χρησιμοποιεί το Service Bus, ώστε να μπορεί να συνδεθεί σε αυτό σύμφωνα με τις προδιαγραφές της SOA.

Λύση

Το **(1)**. Αυτό εξασφαλίζει διαλειτουργικότητα και ευελιξία στην ενσωμάτωση διαφορετικών συστημάτων.

Ερώτημα 8

Η στρατηγική "plan and document" δεν εφαρμόζεται σε ανάπτυξη λογισμικού SaaS.

1. Σωστό.
2. Λάθος.

Λύση

Λάθος. Η στρατηγική "plan and document" μπορεί ακόμα να εφαρμοστεί στην ανάπτυξη λογισμικού SaaS. Παρόλο που το SaaS συχνά επωφελείται από ευέλικτες και επαναληπτικές προσεγγίσεις, ο σωστός σχεδιασμός και η τεκμηρίωση παραμένουν σημαντικά για τη διασφάλιση σαφών απαιτήσεων, τη διατήρηση της συμμόρφωσης και την παροχή αξιόπιστων ενημερώσεων και υποστήριξης.

Ερώτημα 9

Μια εφαρμογή που ακολουθεί την αρχιτεκτονική microservices είναι δυνατόν να προσαρμοστεί ώστε να λειτουργεί σε ένα οικοσύστημα εφαρμογών που ακολουθεί την αρχιτεκτονική SOA.

1. Ναι, με τροποποιήσεις στον μηχανισμό χειρισμού δεδομένων, ώστε να χρησιμοποιεί μια κεντρική υπηρεσία διαχείρισης δεδομένων.
2. Ναι, εφόσον οι τεχνολογίες messaging που χρησιμοποιεί η εφαρμογή microservices είναι συμβατές με αυτές του οικοσυστήματος SOA.
3. Όλες οι υπόλοιπες απαντήσεις είναι σωστές.
4. Ναι, με προσθήκες που αφορούν τη συμμόρφωση με τις απαιτήσεις ανιχνευσιμότητας και συνεργασίας των εφαρμογών στο οικοσύστημα SOA.

Λύση

Το **(3)**.

- Η προσαρμογή μιας εφαρμογής μικρουπηρεσιών σε ένα οικοσύστημα SOA μπορεί να απαιτεί την τροποποίηση του χειρισμού δεδομένων ώστε να ευθυγραμμιστεί με την κεντρική διαχείριση δεδομένων.
- Η διασφάλιση της συμβατότητας των τεχνολογιών ανταλλαγής μηνυμάτων είναι ζωτικής σημασίας για την απρόσκοπτη ενσωμάτωση.
- Η ικανοποίηση των απαιτήσεων SOA για ιχνηλασιμότητα και συνεργασία διασφαλίζει τη σωστή αλληλεπίδραση στο οικοσύστημα.

Ερώτημα 10

Με χρήση Kubernetes μπορούμε να προδιαγράψουμε ένα ελάχιστο αριθμό κόμβων (pods). Τι θα συμβεί εάν ένας από αυτούς τους κόμβους αστοχήσει και ΔΕΝ καταφέρει το σύστημα να τον αντικαταστήσει με έναν νέο;

1. Θα εξακολουθήσει να λειτουργεί το όλο σύστημα, ακόμη και αν υπάρχει μόνο ένας ενεργός κόμβος.
2. Θα διακοπεί προσωρινά η λειτουργία του συστήματος, μέχρι να αντικατασταθεί ο απωλεσθείς κόμβος.

Λύση

Το **(1)**. Εάν ένα από τα pods αποτύχει και το σύστημα δεν καταφέρει να το αντικαταστήσει με ένα νέο, ολόκληρο το σύστημα θα συνεχίσει να λειτουργεί, ακόμη και αν υπάρχει μόνο ένας ενεργός κόμβος. Το Kubernetes έχει σχεδιαστεί για να χειρίζεται τέτοιες αποτυχίες, διατηρώντας τη συνολική λειτουργικότητα του συστήματος όσο τουλάχιστον ένα pod παραμένει λειτουργικό.

Ερώτημα 11

Σε μία αρχιτεκτονική microservices η διαχείριση των δεδομένων εντός κάθε microservice πρέπει να γίνεται:

1. Μέσω οποιασδήποτε από τις υπόλοιπες επιλογές.
2. Μέσω ενός REST API.
3. Με άμεση σύνδεση με το επίπεδο διαχείρισης δεδομένων μέσω κατάλληλου framework.
4. Μέσω ενός middleware ORM.

Λύση

Το **(1)**. Σε μια αρχιτεκτονική μικρουπηρεσιών, η διαχείριση των δεδομένων εντός κάθε μικρουπηρεσίας θα πρέπει να γίνεται μέσω οποιασδήποτε από τις άλλες επιλογές. Οι μικρουπηρεσίες μπορούν να χρησιμοποιούν REST APIs, άμεσες συνδέσεις σε επίπεδα διαχείρισης δεδομένων ή ORMs ενδιάμεσου λογισμικού, ανάλογα με τις συγκεκριμένες απαιτήσεις και τις επιλογές σχεδιασμού του συστήματος.

Ερώτημα 12

Τι επιτυγχάνουμε με την τεχνική του “mocking” στο unit testing;

Λύση

Με την τεχνική mocking στο unit testing, προσομοιώνουμε τη συμπεριφορά πραγματικών αντικειμένων για να απομονώσουμε και να ελέγξουμε συγκεκριμένα στοιχεία του κώδικα. Αυτό μας επιτρέπει να:

- Να δοκιμάζουμε μια μονάδα χωρίς να βασιζόμαστε σε εξωτερικές εξαρτήσεις, όπως βάσεις δεδομένων, API ή άλλες υπηρεσίες, διασφαλίζοντας ότι το περιβάλλον δοκιμής είναι ελεγχόμενο και προβλέψιμο.
- Να εστιάζουμε στη λογική της μονάδας που δοκιμάζεται, παρέχοντας προκαθορισμένες αποκρίσεις από τα αντικείμενα που παρωδούνται.
- Βελτίωση της ταχύτητας και της αποτελεσματικότητας των δοκιμών, καθώς τα αντικείμενα που προσομοιάζουν είναι συνήθως ταχύτερα στη δημιουργία και την εκτέλεση σε σύγκριση με τα πραγματικά αντικείμενα.
- Εντοπισμός και διάγνωση ζητημάτων εντός της συγκεκριμένης μονάδας υπό δοκιμή πιο αποτελεσματικά εξαλείφοντας την επιρροή εξωτερικών παραγόντων.

Ερώτημα 13

Μια εφαρμογή που ακολουθεί την αρχιτεκτονική MVC είναι δυνατόν να προσαρμοστεί ώστε να λειτουργεί σε ένα οικοσύστημα εφαρμογών που ακολουθεί την αρχιτεκτονική SOA:

1. Ναι, με προσθήκες στον "Controller" που αφορούν τη συμμόρφωση με τις απαιτήσεις ανιχνευσιμότητας και συνεργασίας των εφαρμογών στο οικοσύστημα SOA.
2. Ναι, με προσθήκη ενός "View" που θα κάνει τη δουλειά της συμμόρφωσης με τις απαιτήσεις του SOA.
3. Ναι, με τροποποιήσεις στο "Model", ώστε να χρησιμοποιεί μια κεντρική υπηρεσία διαχείρισης δεδομένων στο SOA.
4. Όλες οι υπόλοιπες απαντήσεις είναι σωστές.

Λύση

Το **(4)**.

Στην SOA, οι υπηρεσίες συχνά απαιτούν πρόσθετα χαρακτηριστικά για συμμόρφωση, ιχνηλασιμότητα και συνεργασία. Προσθέτοντας αυτά τα χαρακτηριστικά στο επίπεδο "Controller" της εφαρμογής MVC, εξασφαλίζετε ότι κάθε υπηρεσία μπορεί να καταγράφει δραστηριότητες, να παρακολουθεί συναλλαγές και να αλληλεπιδρά απρόσκοπτα με άλλες υπηρεσίες στο οικοσύστημα. Αυτό θα μπορούσε να περιλαμβάνει την ενσωμάτωση πλαισίων καταγραφής, εργαλείων παρακολούθησης και πρωτοκόλλων επικοινωνίας μεταξύ υπηρεσιών που ευθυγραμμίζονται με τα πρότυπα SOA.

Ενώ η "View" στην MVC αφορά συνήθως τη λογική παρουσίασης, σε ένα πλαίσιο SOA, μπορεί να χρειαστεί να συμμορφώνεται με συγκεκριμένα πρότυπα ή μορφές που απαιτούνται από το περιβάλλον SOA. Αυτό θα μπορούσε να περιλαμβάνει τη διασφάλιση ότι η "View" είναι ικανή να αποδίδει σωστά τα δεδομένα που λαμβάνονται από υπηρεσίες SOA ή να χρησιμοποιεί τυποποιημένα πρότυπα και στυλ που είναι συνεπή σε

όλο το οικοσύστημα. Παρόλο που αυτό δεν είναι τόσο κρίσιμο όσο άλλες προσαρμογές, εξασφαλίζει ομοιομορφία και συνέπεια στο επίπεδο παρουσίασης της εφαρμογής.

Το "Model" στο MVC αναπαριστά τα δεδομένα και την επιχειρησιακή λογική. Σε ένα οικοσύστημα SOA, αυτό το στρώμα πρέπει συχνά να αλληλεπιδρά με μια κεντρική υπηρεσία διαχείρισης δεδομένων, ώστε να διασφαλίζεται η συνέπεια και η ακεραιότητα των δεδομένων στις διάφορες υπηρεσίες. Η τροποποίηση του "Model" ώστε να χρησιμοποιεί μια τέτοια κεντρική υπηρεσία το ευθυγραμμίζει με την έμφαση της SOA στην κοινή διαχείριση δεδομένων. Αυτό θα μπορούσε να περιλαμβάνει την αλλαγή του τρόπου πρόσβασης και αποθήκευσης των δεδομένων, διασφαλίζοντας ότι το " Model" μπορεί να επικοινωνεί με κεντρικές βάσεις δεδομένων ή άλλες υπηρεσίες δεδομένων που παρέχονται από την υποδομή SOA.

Ερώτημα 14

Μια κατανεμημένη (distributed) βάση δεδομένων (distributed DBMS) είναι δυνητικά τμήμα μιας εφαρμογής που υλοποιείται με την αρχιτεκτονική microservices.

1. Ναι, εφόσον κάθε microservice διατηρεί την ευθύνη διαχείρισης και συγχρονισμού των δεδομένων που χειρίζεται, ανεξάρτητα από την φυσική κατανομή στη ΒΔ.
2. Ναι, διότι ο συγχρονισμός γίνεται με ευθύνη του distributed DBMS, γεγονός που απλοποιεί την υλοποίηση των microservices.
3. Όχι, κάθε microservice πρέπει να έχει τη δική του ΒΔ.

Λύση

Το **(1)**.

Ερώτημα 15

Ο πηγαίος κώδικας που παράγεται αυτόματα και ενσωματώνεται σε μια εφαρμογή SaaS:

1. Πρέπει να αποφεύγεται γιατί συνήθως είναι "φλύαρος".
2. Είναι προτιμότερος γιατί είναι πιο εύκολο να συντηρηθεί.
3. Είναι προτιμότερος γιατί κάνει καλύτερη διαχείριση πόρων.
4. Είναι προτιμότερος γιατί είναι γρηγορότερος.

Λύση

Το **(2)**. Ο αυτόματα παραγόμενος πηγαίος κώδικας σε μια εφαρμογή SaaS είναι προτιμότερος επειδή είναι ευκολότερο να συντηρηθεί. Παρόλο που μπορεί μερικές φορές να είναι μακροσκελής, η πτυχή της συντηρησιμότητας γενικά υπερτερεί άλλων εκτιμήσεων, ιδίως σε περιβάλλοντα όπου οι γρήγορες ενημερώσεις και η συνεπής λειτουργικότητα είναι κρίσιμες.

Ερώτημα 16

Τι από τα παρακάτω μπορεί να είναι διαφορετικό μεταξύ του περιβάλλοντος εγκατάστασης (deployment) του προγραμματιστή (developer) και του περιβάλλοντος της παραγωγής (production);

1. Όλες οι υπόλοιπες απαντήσεις είναι σωστές.

2. Η ύπαρξη ενδιάμεσων επιπέδων cache.
3. Καμία από τις υπόλοιπες απαντήσεις δεν είναι σωστή.
4. Η βάση δεδομένων.
5. Το λειτουργικό σύστημα.

Λύση

Το **(1)**. Όλες οι άλλες απαντήσεις είναι σωστές. Οι διαφορές μεταξύ του περιβάλλοντος ανάπτυξης του προγραμματιστή και του περιβάλλοντος παραγωγής μπορεί να περιλαμβάνουν την ύπαρξη ενδιάμεσων επιπέδων προσωρινής αποθήκευσης, τη βάση δεδομένων και το λειτουργικό σύστημα. Αυτές οι διαφοροποιήσεις μπορεί να επηρεάσουν τη συμπεριφορά και την απόδοση της εφαρμογής.

Ερώτημα 17

Διαλειτουργικότητα διοδίων ως SaaS.

Τα ακόλουθα ηλεκτρονικά συστήματα διέλευσης διοδίων διαλειτουργούν (δηλαδή επιτρέπουν την χρήση όλων των πομποδεκτών σε όλα τα δίκτυα), από τις 4.11.2020: aodos.gr, gefyra.gr, egnatia.eu, kentrikiodos.gr, moreas.com.gr, neaodos.gr, olympiaodos.gr. Υποθέτουμε τα εξής (που δεν απέχουν και πολύ από την πραγματικότητα):

Δεν χρησιμοποιούν όλα την ίδια εφαρμογή διαχείρισης διελεύσεων.

Όλα υλοποιούν, το καθένα με δικό του τρόπο, τις λειτουργίες "provider" (δήλωση σε ποιον πάροχο ανήκει ένας πομποδέκτης), "load" (φόρτωση πομποδέκτη με χρήματα), "pass" (διέλευση) και "balance" (υπόλοιπο χρημάτων στον πομποδέκτη).

Θέλουμε να παρέχουμε ως SaaS την υπηρεσία διαλειτουργικότητας. Προτείνετε μια αρχιτεκτονική για την εφαρμογή μας, που να βασίζεται στις αρχές SOA (αν ο ΑΜ σας τελειώνει σε άρτιο αριθμό) ή στις αρχές microservices (αν ο ΑΜ σας τελειώνει σε περιττό αριθμό). Παραδώστε ένα διάγραμμα UML component με τα στοιχεία της εφαρμογής μας, συνοδευόμενο από πολύ σύντομο σχολιασμό.

Λύση

Παράδειγμα διαγράμματος συνιστωσών UML για αρχιτεκτονική μικρουπηρεσιών:

[Υπηρεσία διαλειτουργικότητας] --|> [Υπηρεσία παρόχου]

--|> [Υπηρεσία φόρτωσης]

--|> [Pass Service]

--|> [Υπηρεσία ισορροπίας]

- Υπηρεσία διαλειτουργικότητας: Λειτουργεί ως ενορχηστρωτής που συντονίζει τις αλληλεπιδράσεις μεταξύ διαφόρων συστημάτων διέλευσης διοδίων.
- Υπηρεσία παρόχου: Διαχειρίζεται τις δηλώσεις των παρόχων πομποδεκτών.
- Υπηρεσία φόρτωσης: Χειρίζεται τη φόρτωση χρημάτων σε πομποδέκτες.

- Υπηρεσία διέλευσης: Διαχειρίζεται τις λειτουργίες διαμετακόμισης σε διαφορετικά συστήματα διοδίων.
- Υπηρεσία ισοζυγίου: Ανάκτηση του χρηματικού υπολοίπου των πομποδεκτών.

Ερώτημα 18

Το σημαντικότερο σημείο προσοχής στην ανάπτυξη μιας εφαρμογής που θα διατεθεί ως SaaS είναι:

1. Κάτι άλλο που δεν συμπεριλαμβάνεται στις υπόλοιπες απαντήσεις.
2. Να τρέχει σε όλους τους browser.
3. Να χειρίζεται σωστά θέματα ασφάλειας ευαίσθητων δεδομένων.
4. Να υλοποιεί σωστά την επιλεγμένη αρχιτεκτονική (MVC, microservices, κ.λπ.).

Λύση

Το **(3)**. Το σημαντικότερο σημείο προσοχής κατά την ανάπτυξη μιας εφαρμογής που θα παραδοθεί ως SaaS είναι ο σωστός χειρισμός των θεμάτων ασφάλειας ευαίσθητων δεδομένων. Η διασφάλιση της ασφάλειας των δεδομένων είναι ζωτικής σημασίας για την προστασία των πληροφοριών των χρηστών, τη συμμόρφωση με τους κανονισμούς και τη διατήρηση της εμπιστοσύνης στην υπηρεσία.

Ερώτημα 19

Ένα docker container παρέχει υπηρεσίες ανάλογες του:

1. IaaS.
2. SaaS.
3. PaaS ή SaaS.
4. IaaS ή PaaS.
5. SaaS ή PaaS ή IaaS.
6. PaaS.

Λύση

Το **(6)**. Ένα Docker container παρέχει υπηρεσίες παρόμοιες με το PaaS (Platform as a Service). Τα Docker containers ενθυλακώνουν τις εφαρμογές και τις εξαρτήσεις τους, παρέχοντας ένα περιβάλλον για τους προγραμματιστές να κατασκευάζουν, να αποστέλλουν και να εκτελούν εφαρμογές χωρίς να ανησυχούν για την υποκείμενη υποδομή.

Ερώτημα 20

Κατά τη σχεδίαση μιας εφαρμογής που προορίζεται να διατεθεί ως SaaS με την agile μέθοδο:

1. Είναι καλό να ακολουθούμε μια ευέλικτη (agile) στρατηγική σε όλη την έκταση της σχεδίασης που επιτρέπει οποιαδήποτε μεταβολή απαιτηθεί.
2. Όλες οι υπόλοιπες απαντήσεις είναι σωστές.
3. Αρκεί να καταγράφουμε τι έχουμε κάνει σε κάποιο κείμενο που είναι εύκολο να το ενημερώνουμε αν κάνουμε μεταβολές.
4. Καμία από τις υπόλοιπες απαντήσεις δεν είναι σωστή.

5. Είναι καλό να χρησιμοποιούμε εργαλεία και πρότυπα.

Λύση

Το **(2)**. Όταν σχεδιάζετε μια εφαρμογή που προορίζεται να παραδοθεί ως SaaS με ευέλικτο τρόπο, είναι καλό να ακολουθείτε μια ευέλικτη στρατηγική καθ' όλη τη διάρκεια του σχεδιασμού που επιτρέπει κάθε απαιτούμενη αλλαγή, να χρησιμοποιείτε εργαλεία και πρότυπα και να καταγράφετε τι έχει γίνει σε ένα έγγραφο που είναι εύκολο να ενημερωθεί αν γίνουν αλλαγές. Όλες αυτές οι πρακτικές εξασφαλίζουν ευελιξία, τεκμηρίωση και αποτελεσματική διαχείριση του έργου.

Ερώτημα 21

Μία εφαρμογή που χρησιμοποιεί την αρχιτεκτονική SOA:

1. Όλες οι υπόλοιπες απαντήσεις είναι σωστές.
2. Καμία από τις υπόλοιπες απαντήσεις δεν είναι σωστή.
3. Χρησιμοποιεί το πρωτόκολλο SOAP για επικοινωνία με το Service Bus.
4. Έχει πάντα τουλάχιστον ένα επίπεδο επικοινωνίας με τον χρήστη (frontend).
5. Έχει πάντα ένα επίπεδο διαχείρισης δεδομένων.

Λύση

Το **(2)**. SOAP δεν σχετίζεται πάντα με SOA. Υπάρχουν περιπτώσεις όπου δεν απαιτείται fronted (π.χ. στο μετρό που χτυπάς το εισιτήριο). Μπορεί να έχει παραπάνω από 1 επίπεδο διαχείρισης δεδομένων.

Ερώτημα 22

Οι περιορισμοί που επιβάλλονται από τις επικεφαλίδες (headers) CORS μπορεί να υλοποιηθούν και με άλλο τρόπο.

1. Ναι, σε μερικές περιπτώσεις.
2. Όχι.
3. Ναι, σε όλες τις περιπτώσεις.

Λύση

Το **(1)**. Οι εναλλακτικές μέθοδοι περιλαμβάνουν τη χρήση του JSONP για την ανάκτηση δεδομένων ή τους πληρεξούσιους στην πλευρά του διακομιστή για τον χειρισμό των cross-origin αιτήσεων, αν και αυτές οι προσεγγίσεις έχουν τους δικούς τους περιορισμούς και ζητήματα ασφάλειας.

Ερώτημα 23

Μία εφαρμογή που ακολουθεί την αρχιτεκτονική microservices είναι δυνητικά περισσότερο ανθεκτική σε απώλεια δικτυακών συνδέσεων μεταξύ των microservices, απ' ότι μια εφαρμογή SOA.

1. Ναι, αλλά χωρίς την πλήρη λειτουργικότητά της.

2. Όχι, η απώλεια δικτυακών συνδέσεων είναι "μοιραία" για τη λειτουργικότητα της εφαρμογής ανεξάρτητα από την αρχιτεκτονική.
3. Ναι, δυνητικά με την πλήρη λειτουργικότητά της.

Λύση

Το **(1)**. Οι μικρουπηρεσίες είναι σχεδιασμένες να διαχειρίζονται μερικές αποτυχίες και να συνεχίζουν να λειτουργούν, αν και ορισμένα χαρακτηριστικά μπορεί να είναι υποβαθμισμένα ή μη διαθέσιμα μέχρι να αποκατασταθούν οι συνδέσεις.

Ερώτημα 24

Η απαίτηση μιας ευέλικτης μεθοδολογίας ανάπτυξης λογισμικού για "συνεχή παροχή προς τον πελάτη λογισμικού που λειτουργεί", αναδεικνύει την εφαρμογή μίας από τις ακόλουθες αρχές της ευέλικτης ανάπτυξης ως μεγαλύτερη πρόκληση σε σχέση με τις άλλες:

1. Σύντομοι σε διάρκεια κύκλοι ανάπτυξης (sprints).
2. Καθορισμός της αρχιτεκτονικής από αυτο-οργανούμενες ομάδες.
3. Συνεχής προσήλωση στην τεχνική αριστεία και την καλή σχεδίαση.
4. Αποδοχή των μεταβολών στις απαιτήσεις του πελάτη σε οποιαδήποτε φάση.

Λύση

Το **(1)**. Η απαίτηση "να παρέχει συνεχώς στον πελάτη λογισμικό που λειτουργεί" καθιστά τους μικρότερους κύκλους ανάπτυξης (sprints) πιο δύσκολους. Η διασφάλιση της συχνής παράδοσης λειτουργικού λογισμικού απαιτεί ισχυρές πρακτικές σχεδιασμού, δοκιμών και ολοκλήρωσης για τη διατήρηση της ποιότητας και της λειτουργικότητας.

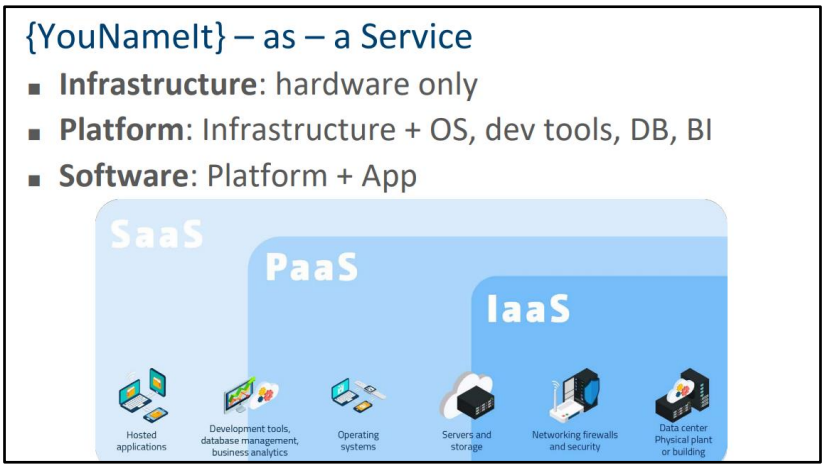
Ερώτημα 25

Η παροχή υπηρεσιών φιλοξενίας μιας ιστοσελίδας (π.χ. apache, PHP, MySQL) σε διαμοιραζόμενο server θεωρείται:

1. PaaS.
2. Καμία από τις άλλες απαντήσεις δεν είναι σωστή:
3. IaaS.
4. SaaS.

Λύση

Το **(1)**. Από τις διαφάνειες:



Ερώτημα 26

Το σημαντικότερο πλεονέκτημα της χρήσης του υπολογιστικού νέφους για τη φιλοξενία εφαρμογών SaaS είναι η:

1. Διαθεσιμότητα.
2. Ταχύτητα.
3. Οικονομική χρέωση.
4. Χρέωση ανάλογα με την κατανάλωση.
5. Ασφάλεια.

Λύση

Το **(1)**. Το σημαντικότερο πλεονέκτημα της χρήσης του υπολογιστικού νέφους για τη φιλοξενία εφαρμογών SaaS είναι η διαθεσιμότητα. Οι πάροχοι υπολογιστικού νέφους προσφέρουν συνήθως υψηλή διαθεσιμότητα, πλεονασμό και εγγυήσεις διαθεσιμότητας, διασφαλίζοντας ότι οι εφαρμογές SaaS παραμένουν προσβάσιμες και αξιόπιστες για τους χρήστες.

Ερώτημα 27

Με τη μέθοδο του test-driven development:

1. Τα τεστ γράφονται πριν γραφεί ο κώδικας που υλοποιεί κάποια λειτουργικότητα.
2. Οι προγραμματιστές γράφουν τεστ αντί για κώδικα.
3. Ο κώδικας γράφεται πρώτα και μετά ελέγχεται διεξοδικά με μικρά τεστ.
4. Τα τεστ και ο κώδικας γράφονται από προγραμματιστές που εργάζονται σε ζεύγη: ένας γράφει τον κώδικα και ο άλλος τα αντίστοιχα τεστ.

Λύση

Το **(1)**. Η προσέγγιση αυτή διασφαλίζει ότι ο κώδικας ανταποκρίνεται στην απαιτούμενη λειτουργικότητα και βοηθά στην έγκαιρη αντιμετώπιση προβλημάτων κατά τη διαδικασία ανάπτυξης.

Ερώτημα 28

Αρχιτεκτονική microservices και συστήματα οικονομικών συναλλαγών (π.χ. πληρωμών):

1. "Παίζει", εφόσον η διαχείριση των transactions γίνεται από ένα microservice και μόνο.
2. "Παίζει", αλλά απαιτείται κατάλληλη υλοποίηση η οποία να υποστηρίζει transactions.
3. Όλες οι υπόλοιπες απαντήσεις είναι σωστές.
4. "Παίζει" καλύτερα για ασύγχρονα transactions (μη πραγματικού χρόνου).

Λύση

Το **(2)**. Η αρχιτεκτονική μικρουπηρεσιών μπορεί να είναι κατάλληλη για συστήματα χρηματοοικονομικών συναλλαγών, αλλά απαιτεί κατάλληλη υλοποίηση που να υποστηρίζει συναλλαγές. Η διασφάλιση της συνέπειας και της ακεραιότητας των δεδομένων σε κατανεμημένα συστήματα είναι πολύπλοκη και απαιτεί ισχυρούς μηχανισμούς για τη διαχείριση των συναλλαγών.

Ερώτημα 29

Με χρήση Kubernetes μπορούμε να υλοποιήσουμε υπηρεσίες Enterprise Service Bus σε μια αρχιτεκτονική SOA.

1. Σωστό.
2. Λάθος.

Λύση

Σωστό. Χρησιμοποιώντας το Kubernetes, μπορούμε να υλοποιήσουμε υπηρεσίες Enterprise Service Bus (ESB) σε μια αρχιτεκτονική SOA. Το Kubernetes μπορεί να διαχειριστεί και να ενορχηστρώσει την ανάπτυξη διαφόρων μικρουπηρεσιών που παρέχουν συλλογικά λειτουργίες ESB, όπως υπηρεσίες ανταλλαγής μηνυμάτων, δρομολόγησης και ολοκλήρωσης.

Επαναληπτική 21

Ερώτημα 4

Η παροχή υποδομών δικτύου και ασφάλειας ως υπηρεσία θεωρείται:

1. Virtualization.
2. SaaS.
3. IaaS.
4. PaaS.

Λύση

Το **(3)**.

Ερώτημα 5

Η στρατηγική "plan and document" δεν είναι δυνατό να εφαρμοστεί σε ανάπτυξη λογισμικού SaaS.

1. Σωστό
2. Λάθος

Λύση

Λάθος. Παρόμοιο με K21, E8.

Ερώτημα 6

Ο πηγαίος κώδικας που παράγεται αυτόματα και ενσωματώνεται σε μια εφαρμογή SaaS:

1. Είναι προτιμότερος γιατί κάνει καλύτερη διαχείριση πόρων.
2. Πρέπει να αποφεύγεται γιατί συνήθως είναι "φλύαρος".
3. Είναι προτιμότερος γιατί είναι πιο εύκολο να συντηρηθεί.
4. Είναι προτιμότερος γιατί είναι γρηγορότερος.

Λύση

Το **(3)**. Παρόμοιο με K21, E15.

Ερώτημα 7

Η χρήση διακριτών διεπαφών (interfaces) μεταξύ συστατικών στοιχείων λογισμικού, όπως π.χ. REST:

1. Διευκολύνει την υλοποίηση κληρονομικότητας κατ' αντιστοιχία με το αντικειμενοστρεφές παράδειγμα προγραμματισμού.
2. Διευκολύνει την επαναχρησιμοποίηση λογισμικού με αντίβαρο τη μη βέλτιστη χρήση πόρων.
3. Καμία από τις υπόλοιπες απαντήσεις δεν είναι σωστή.

4. Μπορεί να γίνει μόνο για συγκεκριμένα πρότυπα αρχιτεκτονικής λογισμικού.

Λύση

Το **(2)**. Παρόμοιο με K21, E5.

Ερώτημα 8

Το σημαντικότερο πλεονέκτημα της χρήσης του υπολογιστικού νέφους για τη φιλοξενία εφαρμογών SaaS είναι:

1. Η ταχύτητα και οι υποστηριζόμενες αρχιτεκτονικές.
2. Η χρέωση ανάλογα με την κατανάλωση πόρων και η διαθεσιμότητα.
3. Η διαθεσιμότητα και η ασφάλεια.

Λύση

Το **(3)**. Παρόμοιο με K21, E26.

Ερώτημα 9

Από τα παρακάτω αναφερόμενα, αυτό στο οποίο πρέπει να δοθεί ιδιαίτερη προσοχή κατά την ανάπτυξη μιας εφαρμογής SaaS είναι:

1. Να χειρίζεται σωστά θέματα ασφάλειας ευαίσθητων δεδομένων.
2. Κάτι άλλο που δεν συμπεριλαμβάνεται στις υπόλοιπες απαντήσεις.
3. Να τρέχει σε όλους τους browser.
4. Να υλοποιεί σωστά την επιλεγμένη αρχιτεκτονική (MVC, microservices, κ.λπ.).

Λύση

Το **(1)**. Παρόμοιο με K21, E18.

Ερώτημα 10

Ένα από τα παρακάτω **δεν** είναι χαρακτηριστικό μιας ευέλικτης μεθοδολογίας ανάπτυξης λογισμικού:

1. Οι ομάδες ανάπτυξης είναι πιο αποδοτικές όταν δεν υπάρχουν αυστηρά καθορισμένοι ρόλοι.
2. Η αρχιτεκτονική πρέπει να μπορεί να μεταβάλλεται οποτεδήποτε, ακόμη και στο τέλος της ανάπτυξης.
3. Ο πελάτης μεταφέρει ο ίδιος τις απαιτήσεις του στην ομάδα ανάπτυξης.
4. Οι απαιτήσεις του πελάτη μπορούν να μεταβάλλονται ακόμη και λίγο πριν την ολοκλήρωση του έργου.

Λύση

Το **(2)**. Το χαρακτηριστικό που δεν είναι χαρακτηριστικό της ευέλικτης μεθοδολογίας ανάπτυξης λογισμικού είναι ότι "η αρχιτεκτονική πρέπει να μπορεί να αλλάξει ανά πάσα στιγμή, ακόμη και στο τέλος της ανάπτυξης". Ενώ η ευέλικτη δίνει έμφαση στην ευελιξία και την ανταπόκριση στις αλλαγές, η πραγματοποίηση σημαντικών αρχιτεκτονικών αλλαγών στο τέλος της ανάπτυξης δεν ενδείκνυται γενικά, καθώς μπορεί να εισάγει κινδύνους και αστάθεια.

Ερώτημα 11

Η τεχνική κατάτμησης της λειτουργικότητας μιας εφαρμογής σε μικρά τμήματα τα οποία περιλαμβάνουν τις εργασίες "προδιαγραφή > πλάνο > σχεδίαση > ανάπτυξη > έλεγχος > τεκμηρίωση" και παραδίδονται στον πελάτη σταδιακά, εφαρμόζεται ΜΟΝΟ στην ευέλικτη ανάπτυξη λογισμικού.

1. Σωστό.
2. Λάθος.

Λύση

Λάθος. Η τεχνική της τμηματοποίησης της λειτουργικότητας μιας εφαρμογής σε μικρά τμήματα, τα οποία περιλαμβάνουν εργασίες όπως "προδιαγραφές > σχέδιο > σχεδιασμός > ανάπτυξη > δοκιμές > τεκμηρίωση" και παράδοση στον πελάτη σε στάδια, δεν είναι αποκλειστικότητα της ευέλικτης ανάπτυξης λογισμικού. Η προσέγγιση αυτή χρησιμοποιείται επίσης στις επαναληπτικές και αυξητικές μεθοδολογίες.

Ερώτημα 12

Η συμβατότητα (δηλαδή η επαλήθευση της εκδήλωσης της αναμενόμενης συμπεριφοράς) του browser σε μια εφαρμογή SaaS πρέπει να διασφαλίζεται:

1. Ανεξάρτητα από το αν χρησιμοποιείται server ή client-side scripting.
2. Όταν χρησιμοποιείται client-side scripting.
3. Όταν χρησιμοποιείται server-side scripting.

Λύση

Το **(1)**. Και οι δύο τύποι scripting μπορούν να επηρεάσουν τον τρόπο με τον οποίο η εφαρμογή εκτελείται και συμπεριφέρεται σε διαφορετικούς browsers.

Ερώτημα 13

Η αρχιτεκτονική SOA αφορά εφαρμογές που:

1. Τα στοιχεία (components) που τις αποτελούν πρέπει να συνδέονται εσωτερικά μεταξύ τους μέσω υπηρεσιών messaging.
2. Όλες οι υπόλοιπες απαντήσεις είναι σωστές.
3. Έχουν μόνο έναν κόμβο διαχείρισης δεδομένων (πχ DBMS).

4. Μπορούν να ανταλλάσσουν υπηρεσίες με άλλες εφαρμογές μέσω υπηρεσιών messaging.

Λύση

Το **(4)**. Αυτό το χαρακτηριστικό αναδεικνύει την υπηρεσιοκεντρική φύση της SOA, όπου οι υπηρεσίες σχεδιάζονται έτσι ώστε να είναι διαλειτουργικές και επαναχρησιμοποιήσιμες σε διαφορετικά συστήματα και εφαρμογές.

Ερώτημα 14

Μια εφαρμογή που χρησιμοποιεί την αρχιτεκτονική SOA:

1. Όλες οι υπόλοιπες απαντήσεις είναι σωστές.
2. Έχει ακριβώς ένα επίπεδο διαχείρισης δεδομένων.
3. Καμία από τις υπόλοιπες απαντήσεις δεν είναι σωστή.
4. Έχει υποχρεωτικά τουλάχιστον μία διεπαφή με τον χρήστη (frontend).
5. Χρησιμοποιεί υποχρεωτικά το πρωτόκολλο SOAP για επικοινωνία με το Service Bus.

Λύση

Το **(3)**. Παρόμοιο με K21, E21.

Ερώτημα 15

Μια εφαρμογή που ακολουθεί την αρχιτεκτονική MVC είναι δυνατόν να προσαρμοστεί ώστε να λειτουργεί σε ένα οικοσύστημα εφαρμογών που ακολουθεί την αρχιτεκτονική SOA.

1. Ναι, με προσθήκες στον "Controller", οι οποίες αφορούν τη συμμόρφωση με τις απαιτήσεις ανιχνευσιμότητας και συνεργασίας των εφαρμογών στο οικοσύστημα SOA.
2. Ναι, με προσθήκη ενός "View" που θα κάνει τη δουλειά της συμμόρφωσης με τις απαιτήσεις του SOA.
3. Ναι, με τροποποιήσεις στο "Model", ώστε να χρησιμοποιεί μια κεντρική υπηρεσία διαχείρισης δεδομένων στο SOA.

Λύση

Το **(3)**. Παρόμοιο με K21, E13. Εκεί είχε επιλογή «Όλα», όπου αυτή ήταν. Το πιο σημαντικό είναι το Model, και το View το λιγότερο σημαντικό.

Ερώτημα 16

Μια αρχιτεκτονική microservices είναι προτιμότερο να υλοποιείται με choreographer παρά με orchestrator.

1. Όχι, ο orchestrator είναι πάντα πιο ευέλικτος.
2. Ναι, σε κάθε περίπτωση.
3. Στη γενική περίπτωση δεν μπορούμε να αποφανθούμε.

Λύση

Το **(3)**. Στη γενική περίπτωση, δεν μπορούμε να αποφασίσουμε οριστικά αν μια αρχιτεκτονική μικροϋπηρεσιών είναι καλύτερο να υλοποιηθεί με έναν χορογράφο ή έναν ενορχηστρωτή. Η επιλογή εξαρτάται από τις συγκεκριμένες απαιτήσεις και τα χαρακτηριστικά του συστήματος που αναπτύσσεται. Και οι δύο προσεγγίσεις έχουν τα πλεονεκτήματά τους και τα αντισταθμιστικά τους οφέλη.

Ερώτημα 17

Ο choreographer σε μια αρχιτεκτονική microservices είναι "single point of failure", δηλαδή αν "πέσει", τότε η εφαρμογή δεν λειτουργεί.

1. Όχι, εφόσον η ίδια η υλοποίηση του choreographer είναι ανθεκτική με κόμβους έτοιμους να αναλάβουν.
2. Όταν ο βασικός κόμβος choreographer "πέσει".
3. Ναι, ο choreographer είναι single point of failure και η εφαρμογή πρέπει να είναι όσο πιο ανθεκτική γίνεται στην "απώλειά" του.
4. Όχι, η "απώλεια" του choreographer δεν καθιστά ολόκληρη την εφαρμογή μη-λειτουργική.

Λύση

Το **(1)**. Εάν ο choreographer έχει σχεδιαστεί ώστε να είναι ανθεκτικός και διαθέτει μηχανισμούς εναλλαγής αποτυχίας, η αποτυχία του δεν καθιστά απαραίτητα ολόκληρη την εφαρμογή μη λειτουργική.

Ερώτημα 18

Μια εφαρμογή που ακολουθεί την αρχιτεκτονική microservices σε περίπτωση απώλειας δικτυακών συνδέσεων μεταξύ των microservices:

1. Κινδυνεύει να χάσει τη συνέπεια (consistency) των δεδομένων.
2. Θα έχει χειρότερες επιδόσεις, αλλά χωρίς κίνδυνο απώλειας συνέπειας των δεδομένων.
3. Κινδυνεύει να χάσει τη διαθεσιμότητα της.
4. Καμία από τις υπόλοιπες απαντήσεις δεν είναι σωστή.

Λύση

Το **(1)**. Μια εφαρμογή που ακολουθεί την αρχιτεκτονική μικροϋπηρεσιών κινδυνεύει να χάσει τη συνοχή των δεδομένων σε περίπτωση απώλειας των δικτυακών συνδέσεων μεταξύ των μικροϋπηρεσιών. Οι αποτυχίες του δικτύου μπορούν να διαταράξουν την επικοινωνία και τον συγχρονισμό μεταξύ των υπηρεσιών, οδηγώντας ενδεχομένως σε ασυνέπειες.

Ερώτημα 19

Μια κατανεμημένη (distributed) βάση δεδομένων (distributed DBMS) είναι δυνητικά τμήμα μιας εφαρμογής που υλοποιείται με την αρχιτεκτονική microservices.

1. Όχι, κάθε microservice πρέπει υποχρεωτικά να έχει τη δική του υλοποίηση ΒΔ που να είναι ανεξάρτητη από τις άλλες.
2. Ναι, μόνο εφόσον κάθε microservice διατηρεί την ευθύνη διαχείρισης και συγχρονισμού των δεδομένων που χειρίζεται, ανεξάρτητα από την φυσική κατανομή στη ΒΔ.

Λύση

Το **(2)**. Παρόμοιο με K21, E14.

Ερώτημα 20

Σε μία αρχιτεκτονική microservices η διαχείριση των δεδομένων εντός κάθε microservice πρέπει να γίνεται:

1. Μέσω ενός middleware ORM.
2. Μέσω οποιασδήποτε από τις υπόλοιπες επιλογές.
3. Με άμεση σύνδεση με το επίπεδο διαχείρισης δεδομένων μέσω κατάλληλου framework.
4. Μέσω ενός REST API.

Λύση

Το **(2)**. Παρόμοιο με K21, E11.

Ερώτημα 21

Με χρήση Kubernetes μπορούμε να προδιαγράψουμε ένα ελάχιστο αριθμό κόμβων (pods). Αν ένας από αυτούς τους κόμβους αστοχήσει και **δεν** καταφέρει το σύστημα να τον αντικαταστήσει με έναν νέο, τότε το σύστημα θα εξακολουθήσει να λειτουργεί:

1. Σωστό: θα εξακολουθήσει να λειτουργεί το όλο σύστημα, ακόμη και αν υπάρχει μόνο ένας ενεργός κόμβος.
2. Λάθος: θα διακοπεί προσωρινά η λειτουργία του συστήματος, μέχρι να αντικατασταθεί ο κόμβος που χάθηκε.

Λύση

Σωστό. Παρόμοιο με K21, E10.

Ερώτημα 22

Ποιο από τα παρακάτω **δεν** είναι ένα από τα στάδια του κύκλου του test-driven development;

1. Yellow.
2. Refactor.
3. Red.
4. Green.

Λύση

Το **(1)**. Το κίτρινο δεν είναι ένα από τα στάδια του κύκλου ανάπτυξης με βάση τη δοκιμή (TDD). Ο κύκλος TDD περιλαμβάνει το Κόκκινο (γράψτε μια δοκιμή που αποτυγχάνει), το Πράσινο (γράψτε ακριβώς αρκετό κώδικα για να περάσει η δοκιμή) και το Refactor (βελτιώστε τον κώδικα διατηρώντας τις δοκιμές να περνούν).

Ερώτημα 23

Με τη μέθοδο του test-driven development:

1. Οι προγραμματιστές γράφουν τεστ αντί για κώδικα.
2. Κώδικας γράφεται πρώτα και μετά ελέγχεται διεξοδικά με μικρά τεστ.
3. Τα τεστ και ο κώδικας γράφονται από προγραμματιστές που εργάζονται σε ζεύγη: ένας γράφει τον κώδικα και ο άλλος τα αντίστοιχα τεστ.
4. Τα τεστ γράφονται πριν γραφεί ο κώδικας που υλοποιεί κάποια λειτουργικότητα.

Λύση

Το **(4)**. Παρόμοιο με K21, E27.

Ερώτημα 24

Ποιο από τα παρακάτω δεν ταιριάζει εδώ;

1. Continuous testing.
2. Continuous integration.
3. Continuous function.
4. Continuous delivery.

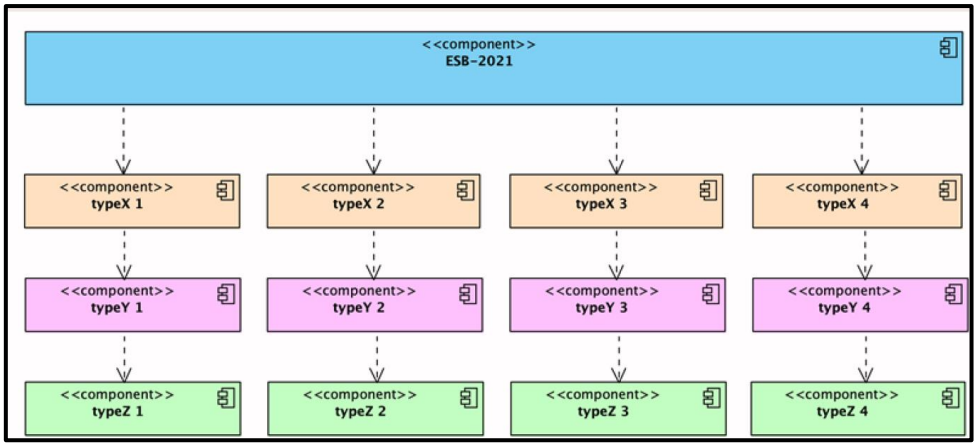
Λύση

Το **(3)**.

- **Continuous testing**: Περιλαμβάνει τη συνεχή δοκιμή του λογισμικού καθ' όλη τη διάρκεια του κύκλου ζωής της ανάπτυξης.
- **Continuous integration**: Περιλαμβάνει τη συχνή συγχώνευση αλλαγών κώδικα σε ένα κεντρικό container, ακολουθούμενη από αυτοματοποιημένες κατασκευές και δοκιμές.
- **Continuous function**: Αυτός ο όρος δεν ταιριάζει στο πλαίσιο των πρακτικών ανάπτυξης λογισμικού που συνήθως συνδέονται με τις σωληνώσεις CI/CD. Δεν αποτελεί τυποποιημένη πρακτική ή μεθοδολογία στην ανάπτυξη λογισμικού.
- **Continuous delivery**: Περιλαμβάνει την αυτόματη ανάπτυξη των αλλαγών κώδικα σε ένα περιβάλλον σταδιοποίησης ή παραγωγής μετά από επιτυχείς δοκιμές.

Ερώτημα 25

Στην παρακάτω αρχιτεκτονική, ονοματίστε τα components typeX, typeY και typeZ:



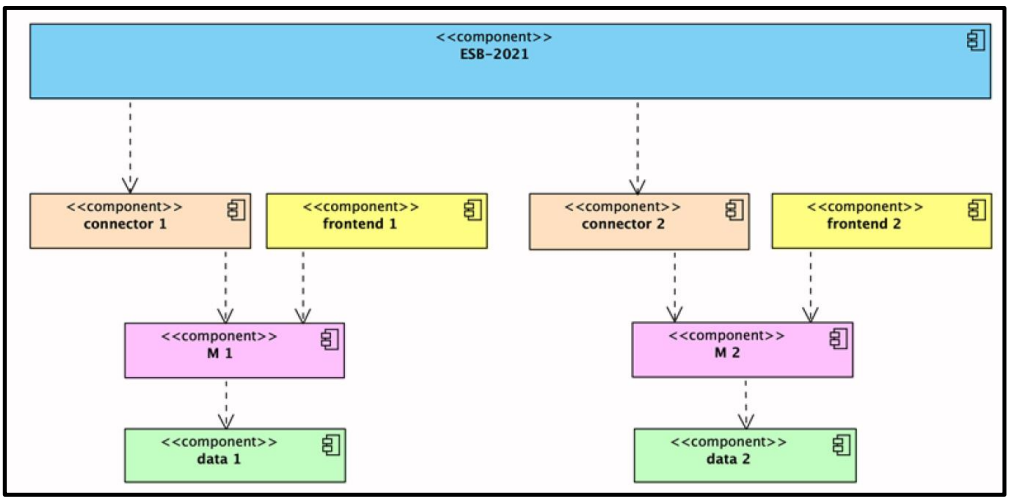
1. TypeX: presentation (frontend), typeY: business logic, typeZ: data layer.
2. TypeX: data layer, typeY: business logic, typeZ: presentation.
3. TypeX: orchestrator, typeY: microservice1, typeZ: microservice2.
4. TypeX: connector, typeY: business logic, typeZ: data layer.
5. TypeX: view, typeY: controller, typeZ: model.

Λύση

Το **(4)**. Δες το σχήμα του επόμενου ερωτήματος. Το ESB συνδέεται με connector, business logic, data layer. Το frontend δεν συνδέεται με το ESB, άρα το X δεν μπορεί να είναι frontend.

Ερώτημα 26

Ένα από τα παρακάτω είναι σωστό για την αρχιτεκτονική του σχήματος:



1. Δεν είναι αρχιτεκτονική SOA διότι τα frontend δεν συνδέονται στο service bus.
2. Είναι μια αρχιτεκτονική SOA με επιμέρους εφαρμογές 3-tier.
3. Είναι μια αρχιτεκτονική SOA με επιμέρους εφαρμογές MVC.
4. Είναι μια αρχιτεκτονική SOA με επιμέρους εφαρμογές microservices.

Λύση

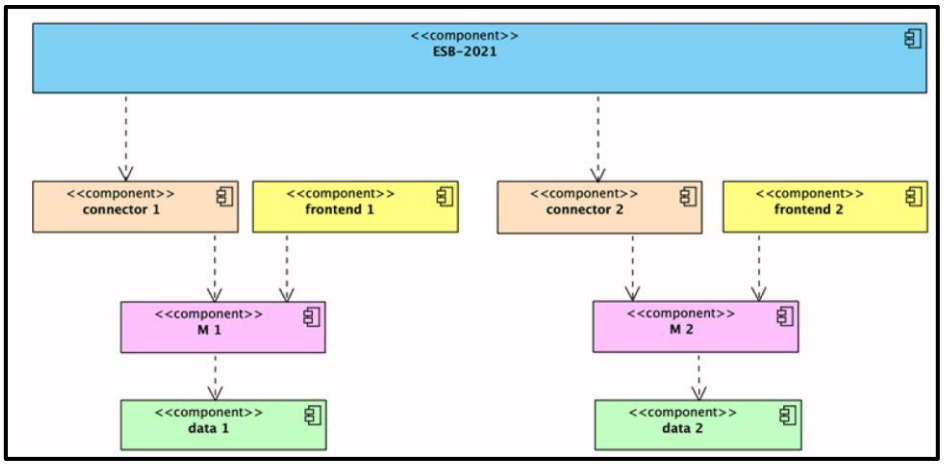
Το **(2)**.

Ανάλυση επιλογών:

1. Λάθος. Δεν πρόκειται για αρχιτεκτονική SOA επειδή το frontend δεν συνδέεται με τον δίαυλο υπηρεσιών. Αυτό είναι εσφαλμένο, διότι τα στοιχεία frontend δεν χρειάζεται να συνδέονται απευθείας με το ESB σε μια αρχιτεκτονική SOA. Μπορούν να συνδεθούν μέσω συνδέσμων.
2. Σωστό. Κάθε μεμονωμένη εφαρμογή έχει τρία επίπεδα:
    a. Connector (βαθμίδα ολοκλήρωσης που συνδέεται με το ESB)
    b. Frontend (βαθμίδα παρουσίασης)
    c. M (μεσαία βαθμίδα ή επιχειρησιακή λογική)
    d. Δεδομένα (βαθμίδα δεδομένων)
3. Λάθος, επειδή το διάγραμμα δεν απεικονίζει ρητά τα στοιχεία MVC. Η εστίαση εδώ είναι στους συνδέσμους, τα frontends και τις βαθμίδες δεδομένων και όχι στο πρότυπο MVC.
4. Λάθος. Ενώ η αρχιτεκτονική θα μπορούσε να ερμηνευθεί ως μικροϋπηρεσίες, τα συγκεκριμένα επίπεδα υποδηλώνουν μια πιο παραδοσιακή δομή 3 επιπέδων παρά μεμονωμένες μικροϋπηρεσίες.

Ερώτημα 27

Στην αρχιτεκτονική του σχήματος, τα M1, M2 φαίνεται να υλοποιούν:



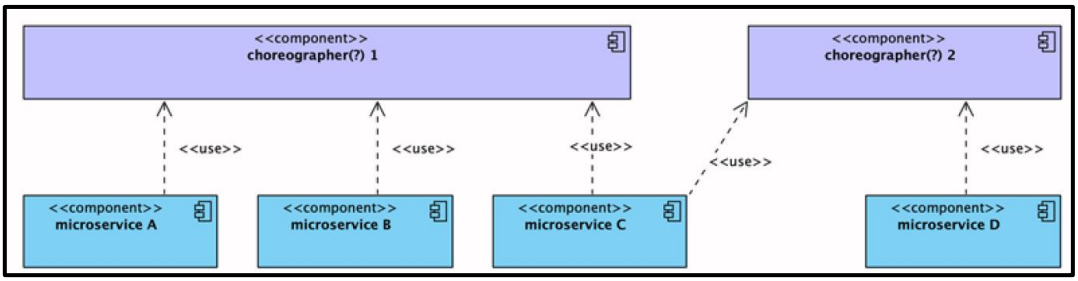
1. Οποιοδήποτε από τα υπόλοιπα.
2. Controller.
3. Business logic.
4. Choreographer.
5. Orchestrator.

Λύση

Το **(3)**.

Ερώτημα 28

Η αρχιτεκτονική που φαίνεται στο παρακάτω σχήμα είναι:



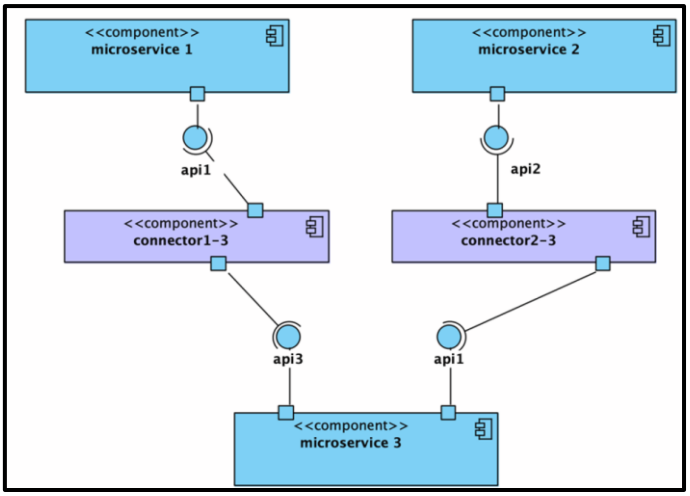
1. Microservices με 2 choreographers.
2. Κάτι άλλο.
3. Microservices, μόνο που τα στοιχεία που χαρακτηρίζονται ως "choreographer(?)" είναι στην πραγματικότητα orchestrators.
4. Δεν είναι microservices διότι δεν επιτρέπονται 2 choreographers.

Λύση

Το **(3)**. Οι orchestrators διαχειρίζονται τον κεντρικό συντονισμό και τις ροές εργασίας μεταξύ των υπηρεσιών, κατευθύνοντάς τες με έναν πιο συγκεντρωτικό τρόπο. Οι connectors στο διάγραμμα φαίνεται να διαχειρίζονται το συντονισμό μεταξύ των μικρουπηρεσιών, υποδεικνύοντας έναν πιο συγκεντρωτικό έλεγχο των αλληλεπιδράσεων.

Ερώτημα 29

Στην αρχιτεκτονική που φαίνεται στο παρακάτω σχήμα, τα components "connector ..." είναι:



1. Choreographers.
2. Service bus.
3. Orchestrators.

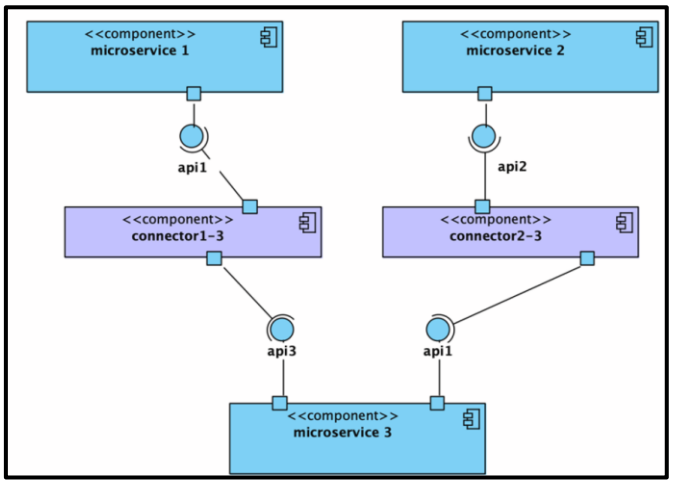
Λύση

Το **(3)**.

Ερώτημα 30

Στην αρχιτεκτονική που φαίνεται στο παρακάτω σχήμα, η χρήση του api1 σε δύο σημεία είναι κακή πρακτική.

1. Ναι.
2. Όχι.

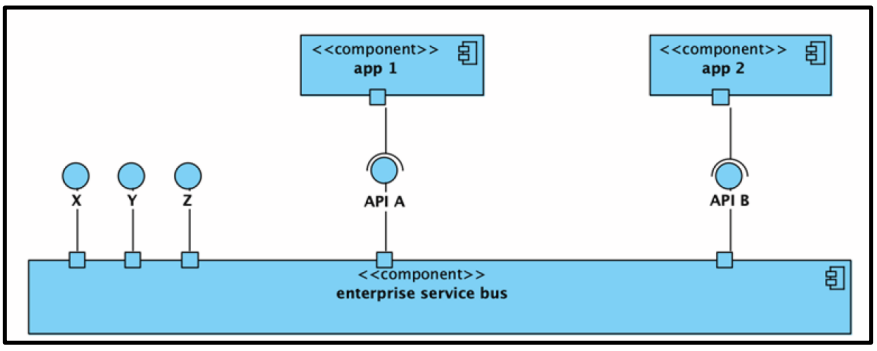


Λύση

Το **(2)**. Η χρήση του «api1» σε δύο σημεία δεν είναι απαραίτητα κακή πρακτική. Εξαρτάται από το πώς έχει σχεδιαστεί το API. Εάν το «api1» έχει σχεδιαστεί ώστε να είναι επαναχρησιμοποιήσιμο και να χειρίζεται σωστά πολλαπλές αλληλεπιδράσεις, η επαναχρησιμοποίησή του μπορεί να προωθήσει τη συνέπεια, να μειώσει τον πλεονασμό και να απλοποιήσει τη συντήρηση. Επομένως, η χρήση του ίδιου API σε πολλαπλούς συνδέσμους ή υπηρεσίες είναι αποδεκτή εφόσον το API υποστηρίζει αυτή τη χρήση.

Ερώτημα 31

Στην ακόλουθη αρχιτεκτονική SOA.



1. Τα X, Y, Z μπορούν και να μην εμφανίζονται στο διάγραμμα εφόσον δεν τα καταναλώνει κανείς

2. Εικονίζονται 3 επιπλέον business-level υπηρεσίες του ESB, τις οποίες δεν "καταναλώνει" καμία εφαρμογή που συνδέεται σε αυτό.
3. Τα X, Y, Z πρέπει να υλοποιούν λειτουργίες service management, discovery, execution και να συνδεθούν σε αυτά τα app 1, app 2.

Λύση

Το **(3)**. Δεν μπορούμε να έχουμε «ξεκρέμαστα» σε SOA.

Κανονική 23

Ερώτημα 1

Η χρήση διαπασών REST σε υπολογιστικά οικοσυστήματα που επιτρέπουν υπηρεσίες web:

i. Διευκολύνει την επαναχρησιμοποίηση υπηρεσιών
ii. Επιτρέπει την υλοποίηση κληρονομικότητας
iii. Μπορεί να γίνει μόνο για συγκεκριμένα πρότυπα σχεδίασης
iv. Μπορεί να γίνει στην αρχιτεκτονική microservices
v. Μπορεί να γίνει στην αρχιτεκτονική SOA
vi. Μειώνει τις επιδόσεις του λογισμικού
vii. Βελτιώνει την ασφάλεια του λογισμικού
viii. Απαιτεί λιγότερους υπολογιστικούς πόρους
ix. Απαιτεί την υλοποίηση όλων των υπηρεσιών με την ίδια τεχνολογία

A. ισχύουν τα iii, iv, v, vii
B. ισχύουν τα i, iv, v, vi
C. ισχύουν τα ii, iv, ix
D. ισχύουν τα i, iv, v, viii

Λύση

B

Ερώτημα 2

Το σημαντικότερο πλεονέκτημα της χρήσης του υπολογιστικού νέφους για τη φιλοξενία εφαρμογών SaaS είναι:

A. Η οικονομική χρέωση
B. Η χρέωση ανάλογα με την κατανάλωση
C. Η διαθεσιμότητα
D. Η ασφάλεια
E. Η ταχύτητα

Λύση

C

Ερώτημα 3

Μια εφαρμογή που ακολουθεί την αρχιτεκτονική microservices είναι δυναμικά περισσότερο ανθεκτική σε απώλεια δικτυακών συνδέσεων μεταξύ των microservices, απ' ότι μια εφαρμογή SOA

A. Είναι λάθος η σύγκριση

B. Ναι, αλλά χωρίς την πλήρη λειτουργικότητά της
C. Ναι, δυναμικά με την πλήρη λειτουργικότητά της
D. Όχι, η απώλεια δικτυακών συνδέσεων είναι "μοιραία" για τη λειτουργικότητα της εφαρμογής ανεξάρτητα από την αρχιτεκτονική

Λύση

C, B

Ερώτημα 4

Μια εφαρμογή που ακολουθεί την αρχιτεκτονική microservices είναι δυνατόν να προσαρμοστεί ώστε να λειτουργεί σε ένα οικοσύστημα εφαρμογών που ακολουθεί την αρχιτεκτονική SOA

A. Με τροποποιήσεις στον μηχανισμό χειρισμού δεδομένων που ακολουθεί το οικοσύστημα
B. Με δεδομένο ότι οι τεχνολογίες messaging που χρησιμοποιεί είναι συμβατές με αυτές του οικοσυστήματος
C. Μόνο οι εφαρμογές που αφορούν τις απαιτήσεις ανεξαρτησίας και συνεργασίας των υπηρεσιών του οικοσυστήματος θα προσαρμοστούν
D. Όλες οι υπόλοιπες προτάσεις
E. Δεν μπορούμε να πούμε

Λύση

D

Ερώτημα 5

Σε μια αρχιτεκτονική SOA η επικοινωνία μεταξύ των εφαρμογών γίνεται

A. Σύγχρονα
B. Είτε σύγχρονα, είτε ασύγχρονα
C. Ασύγχρονα

Λύση

B

Ερώτημα 6

Μια αρχιτεκτονική microservices είναι προτιμότερο να υλοποιείται με choreographer (pub/sub) παρά με orchestrator

A. Δεν μπορούμε να αποφανθούμε για τη γενική περίπτωση
B. Ναι, σε κάθε περίπτωση
C. Όχι, ο choreographer είναι πάντα πιο ευέλικτος

Λύση

A

Ερώτημα 7

Σε αρχιτεκτονική microservices η προσθήκη νέων λειτουργιών (microservices) τα οποία χρησιμοποιούν δεδομένα που ήδη διαθέτει το οικοσύστημα:

A. Είναι ευκολότερη στην περίπτωση υλοποίησης με pub/sub
B. Και τα δύο είναι εφικτά
C. Είναι ευκολότερη στην περίπτωση υλοποίησης με orchestrator
D. Εξαρτάται από την κάθε συγκεκριμένη περίπτωση

Λύση

Όλα

Ερώτημα 8

Σε αρχιτεκτονική microservices μπορούμε να αντικαταστήσουμε:

A. Την υλοποίηση choreographer με ένα orchestrator
B. Και τα δύο είναι εφικτά
C. Μόνο τον choreographer με περισσότερους από έναν orchestrators
D. Εξαρτάται από την κάθε συγκεκριμένη περίπτωση

Λύση

A, C

Ερώτημα 9

Ένα docker container παρέχει υπηρεσίες ανάλογες των υπηρεσιών:

A. Platform-as-a-service
B. Infrastructure-as-a-service
C. Software-as-a-service
D. Κανένα από τα υπόλοιπα
E. Οποιοδήποτε από τα υπόλοιπα

Λύση

A

Ερώτημα 10

Μια εφαρμογή που ακολουθεί την αρχιτεκτονική microservices σε περίπτωση μη λειτουργίας, για οποιαδήποτε λόγο, ενός microservice:

A. Θα σταματήσει να λειτουργεί
B. Θα συνεχίσει τη χρήση των δεδομένων για χάρη της διαθεσιμότητας, αλλά ενδεχομένως με αυξημένες συνέπειες των δεδομένων
C. Αναμένεται ότι θα έχει μεγαλύτερους χρόνους απόκρισης, αλλά χωρίς συνέπειες των δεδομένων
D. Θα συνεχίσει κάποιες από τις υπηρεσίες της, αλλά όχι απαραίτητα τη συνέπεια των δεδομένων
E. Τίποτε από τα υπόλοιπα

Λύση

B

Ερώτημα 11

Αρχιτεκτονική microservices και συστήματα οικονομικών συναλλαγών (πχ πληρωμών):

A. Θα υλοποιηθεί αποκλειστικά και ανάλογα με την περίπτωση
B. "Πακέτο", εφόσον η χρήση των transactions γίνεται από ένα μόνο microservice και μόνο
C. "Πακέτο", εφόσον η χρήση των transactions (μη πραγματικού χρόνου)
D. "Πακέτο", αλλά απαιτείται κατάλληλη υλοποίηση η οποία να υποστηρίζει transactions

---

Λύση

D

Ερώτημα 12

Οι τεχνολογίες messaging μπορούν να χρησιμοποιηθούν:

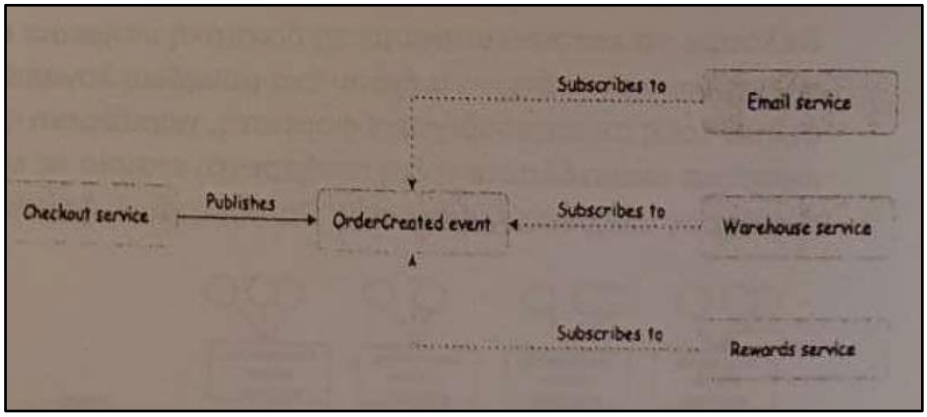
A. Σε αρχιτεκτονικές SOA
B. Σε microservices / orchestrators
C. Σε microservices / choreographers
D. Σε SOA και choreographers
E. Σε όλα τα υπόλοιπα

Λύση

E

Ερώτημα 13

Μια εφαρμογή ηλεκτρονικού εμπορίου έχει διαχωρισμένες λειτουργίες. Ένα UML sequence diagram με τη χρήση των παρακάτω φαίνεται. Η εφαρμογή ακολουθεί την αρχιτεκτονική:



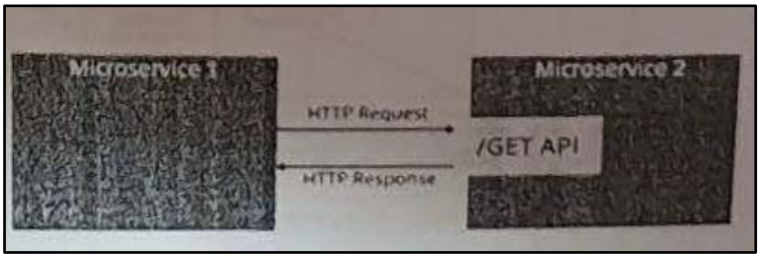
A. SOA
B. Microservices με orchestrator
C. Microservices με choreographer
D. Microservices με SOA

Λύση

C

Ερώτημα 14

Η εικόνα του διπλανού σχήματος θα μπορούσε να είναι UML με χρήση:



A. Δεν μπορούμε να πούμε
B. Microservices με choreographer
C. Microservices με orchestrator

Λύση

B

Sequence Gov.gr Wallet