Εργασία για το Μάθημα

Υπηρεσίες στο Υπολογιστικό Νέφος και την Ομίχλη (ΠΛΗ513)

Ημερομηνία παράδοσης: 9 Ιανουαρίου 2026

Επιμέλεια εργασίας: Προύντζος Αθανάσιος, Τζαβάρας Αιμίλιος

Email επικοινωνίας: aprountzos@tuc.gr, atzavaras@tuc.gr

Ανάπτυξη Συστήματος Διαχείρισης Έργων (Project Management System) με Microservices σε Περιβάλλον Docker

🎯 Σκοπός της Άσκησης

Στόχος της εργασίας είναι η ανάπτυξη ενός **απλοποιημένου συστήματος (web application) τύπου *Jira***, που θα επιτρέπει τη **διαχείριση ομάδων και εργασιών (teams & tasks)** μέσω μιας **microservices αρχιτεκτονικής**.

Το σύστημα θα υποστηρίζει διαφορετικούς **ρόλους χρηστών**, **διαχείριση εργασιών**, **σχόλια**, **αρχεία** και **ειδοποιήσεις**, με έμφαση στη **σωστή αρχιτεκτονική** και **ομαδοποίηση λειτουργιών** σε ανεξάρτητα services.

Η ανάπτυξη θα γίνει με χρήση **Docker** και προτεινόμενα εργαλεία (για βάσεις δεδομένων) την **MySQL** και την **MongoDB**.
Το τελικό αποτέλεσμα θα λειτουργεί ως ένα **απλοποιημένο εργαλείο project management (PMS)**, με βασικές λειτουργίες οργάνωσης έργων, παρόμοιες με εκείνες του Jira ή του Trello.

Σημαντικό: Πρόκειται για μια εργασία που μπορεί να αποτελέσει **ένα πλήρες project για το GitHub σας**, προσφέροντας ένα παραστατικό παράδειγμα πρακτικής εφαρμογής γνώσεων σε backend, frontend και microservices αρχιτεκτονική. Μπορείτε να το χρησιμοποιήσετε ως δείγμα για τις γνώσεις και τις ικανότητές σας (portfolio) για μελλοντικές ευκαιρίες εργασιακής απασχόλησης, κλπ.

🧠 Περιγραφή Εφαρμογής

Η εφαρμογή θα επιτρέπει σε *ομάδες* χρηστών να:

- δημιουργούν και να οργανώνουν έργα (teams/projects),
- διαχειρίζονται εργασίες (tasks),
- αναθέτουν (assign) εργασίες σε μέλη,
- παρακολουθούν την πρόοδο μέσω διαφορετικών σταδίων (TO DO, IN PROGRESS, DONE),
- ανταλλάσσουν σχόλια,
- βλέπουν ειδοποιήσεις (notifications) για αλλαγές ή deadlines που θα εμφανίζονται ***σύγχρονα***, δηλ. απλά θα ανακτώνται από τη βάση και θα προβάλλονται κανονικά στη σελίδα αφού αυτή φορτώσει στον browser, π.χ. σε κάποιο notification dropdown ή όπου αλλού θα είναι εμφανείς/εύχρηστες,
- λαμβάνουν - *προαιρετικά* - ειδοποιήσεις για αλλαγές ή deadlines ***ασύγχρονα (π.χ. με χρήση του μηχανισμού Webhooks ή του Apache Kafka)***, δηλ. θα εμφανίζονται στο χρήστη απευθείας, σε πραγματικό χρόνο (αφού πραγματοποιηθεί κάποια σχετική αλλαγή, π.χ. σχόλιο σε assigned task από άλλο χρήστη) χωρίς ο χρήστης να χρειάζεται να κάνει refresh για να δει τη νέα ειδοποίηση (*προαιρετικό-BONUS feature* για την άσκηση).

Προσοχή: Κάθε χρήστης θα έχει ένα συγκεκριμένο **ρόλο** που καθορίζει τι μπορεί να βλέπει και να επεξεργάζεται.

👥 Ρόλοι Χρηστών

1. Admin (Διαχειριστής Συστήματος)

- Διαχείριση όλων των χρηστών (έγκριση, απενεργοποίηση, αλλαγή ρόλων)
- Δημιουργία και διαγραφή ομάδων
- Προβολή όλων των έργων και εργασιών (χωρίς επεξεργασία)

2. Team Leader (Επικεφαλής Ομάδας)

- Διαχείριση ομάδας (μέλη, πληροφορίες)
- Δημιουργία, επεξεργασία και ανάθεση εργασιών (tasks) σε μορφή απλού κειμένου
- Προσθήκη σχολίων (απλό κείμενο) σε tasks και *προαιρετικά* επισύναψη αρχείων
- Παρακολούθηση της προόδου της ομάδας

3. Team Member (Μέλος Ομάδας)

- Προβολή των ομάδων που ανήκει

- Προβολή και ενημέρωση των δικών του εργασιών
- Προσθήκη σχολίων (και *προαιρετικά* επισύναψη αρχείων) στα tasks του
- Προβολή ειδοποιήσεων για αλλαγές ή προθεσμίες

🧱 Βασικές Οντότητες του Συστήματος

User (Χρήστης)

Ο χρήστης είναι το άτομο που χρησιμοποιεί την εφαρμογή. Μπορεί να είναι διαχειριστής, αρχηγός ομάδας (team leader) ή απλό μέλος. Πρέπει να έχει κάποια βασικά στοιχεία, όπως:

- όνομα χρήστη (username)
- email
- κωδικό πρόσβασης (password)
- όνομα και επώνυμο
- ρόλο (***ADMIN***, ***TEAM LEADER***, ***MEMBER***)
- κατάσταση (αν είναι ήδη ενεργοποιημένος ο χρήστης στο σύστημα ή όχι)

Team (Ομάδα)

Η ομάδα είναι μια μικρή "μονάδα εργασίας" που αποτελείται από χρήστες. Κάθε ομάδα έχει τα δικά της έργα και εργασίες. Πρέπει να έχει:

- όνομα
- περιγραφή
- ημερομηνία δημιουργίας

Task (Εργασία)

Η εργασία είναι κάτι που πρέπει να γίνει μέσα σε μια ομάδα ή ένα έργο. Μπορεί να έχει διαφορετική κατάσταση (π.χ. δεν έχει ξεκινήσει, είναι σε εξέλιξη, έχει ολοκληρωθεί). Πρέπει να περιλαμβάνει:

- τίτλο
- περιγραφή
- ποιος τη δημιούργησε
- σε ποιον έχει ανατεθεί
- κατάσταση (π.χ. *TODO*, *IN PROGRESS*, *DONE*)
- προθεσμία ολοκλήρωσης

- προτεραιότητα (π.χ. χαμηλή, μεσαία, υψηλή)
- ημερομηνία δημιουργίας

Προαιρετικά, μπορεί να περιλαμβάνει και επισυναπτόμενα αρχεία (bonus).

Comment (Σχόλιο)

Το σχόλιο είναι ένα σύντομο μήνυμα ή παρατήρηση που σχετίζεται με μια εργασία. Χρησιμοποιείται για να αφήνουν οι χρήστες σημειώσεις ή ενημερώσεις (π.χ. για την πρόοδο μιας εργασίας). Πρέπει να περιλαμβάνει:

- το κείμενο του σχολίου
- την ημερομηνία που γράφτηκε
- ποιος το έγραψε

Προαιρετικά, μπορεί να περιλαμβάνει και επισυναπτόμενα αρχεία (bonus).

> *Είναι στη διακριτική σας ευχέρεια να επεκτείνετε ή να εμπλουτίσετε τις παραπάνω οντότητες με επιπλέον πεδία ή λειτουργικότητες, ανάλογα με τον τρόπο που επιλέγετε να υλοποιήσετε το σύστημα. Μπορείτε, για παράδειγμα, να προσθέσετε ημερομηνίες ενημέρωσης, εικόνες προφίλ, ετικέτες (tags) στα tasks. Στόχος είναι να σχεδιάσετε ένα* ***λειτουργικό και επεκτάσιμο*** *μοντέλο δεδομένων που να καλύπτει τις ανάγκες ενός απλοποιημένου συστήματος διαχείρισης έργων, παρόμοιο με το Jira.*

Part 1: User Management API

Το **User Management API** ή πιο απλά **User Service (authentication - authorisation)** είναι υπεύθυνο για **όλα όσα σχετίζονται με τους χρήστες** της εφαρμογής. Αυτό περιλαμβάνει τη δημιουργία λογαριασμών (sign up), την είσοδο στο σύστημα (login), την προβολή στοιχείων των χρηστών και τη διαχείριση των ρόλων τους.
Σκοπός του API είναι να διασφαλίζει ότι οι χρήστες έχουν την απαραίτητη δυνατότητα πρόσβασης, τα κατάλληλα δικαιώματα πρόσβασης και ότι τα προσωπικά τους δεδομένα παραμένουν ασφαλή.

Βασικές λειτουργίες του API

- **Δημιουργία νέου χρήστη (Sign Up)**
  Οι χρήστες μπορούν να δημιουργήσουν νέο λογαριασμό παρέχοντας βασικά στοιχεία όπως όνομα χρήστη (username), email, όνομα, επώνυμο και κωδικό πρόσβασης.
  Ο λογαριασμός χρειάζεται **έγκριση από έναν Admin (activation)** προτού ενεργοποιηθεί και επιτραπεί η είσοδος στο σύστημα.

- **Είσοδος στο σύστημα (Login)**
  Το API ελέγχει τα στοιχεία του χρήστη (username και password) και την κατάσταση του χρήστη (ενεργή ή όχι) και, σε περίπτωση επιτυχίας, παρέχει ένα **token ή session** για πρόσβαση στις υπηρεσίες της εφαρμογής.
  Αυτό διασφαλίζει ότι **μόνο** οι εγκεκριμένοι και ενεργοί χρήστες μπορούν να χρησιμοποιούν το σύστημα.

- **Προβολή στοιχείων χρήστη**
  Το API επιτρέπει την ανάκτηση των βασικών πληροφοριών ενός χρήστη, όπως το username, το email, ο ρόλος και η κατάσταση ενεργοποίησης του λογαριασμού.
  Αυτή η λειτουργία μπορεί να χρησιμοποιηθεί από τους ίδιους τους χρήστες ή από διαχειριστές για επίβλεψη.

- **Διαχείριση χρηστών (Admin)**
  Οι διαχειριστές μπορούν να ελέγχουν τη συμμετοχή των χρηστών στο σύστημα.
  Οι βασικές δυνατότητες περιλαμβάνουν:
  - Ενεργοποίηση χρηστών.
  - Αλλαγή ρόλου των χρηστών (team leader, team member).
  - Απενεργοποίηση ή διαγραφή λογαριασμών που δεν χρειάζονται πλέον.

Part 2: Team Management API

Το **Team Management API** ή πιο απλά **Team Service** είναι υπεύθυνο για **τη δημιουργία, προβολή και διαχείριση ομάδων εργασίας** μέσα στην εφαρμογή.
Μέσω αυτού, οι χρήστες μπορούν να οργανώνουν ομάδες, να βλέπουν ποιοι ανήκουν σε κάθε ομάδα και να διαχειρίζονται τις πληροφορίες της ομάδας ανάλογα με τον ρόλο τους.
Σκοπός του API είναι να εξασφαλίζει ότι οι ομάδες είναι καλά οργανωμένες και ότι η διαχείριση τους γίνεται με ασφαλή και ελεγχόμενο τρόπο.

Βασικές λειτουργίες του API

- **Δημιουργία ομάδας (Admin)**
  Οι διαχειριστές μπορούν να δημιουργούν νέες ομάδες παρέχοντας βασικές πληροφορίες όπως όνομα, περιγραφή και αρχηγό της ομάδας (leader).
  Η δημιουργία ομάδας αποτελεί τη βάση για την οργάνωση των εργασιών και των χρηστών μέσα στην εφαρμογή.
- **Επεξεργασία ομάδας (Leader / Admin)**
  Το API επιτρέπει στους αρμόδιους χρήστες να ενημερώνουν τα στοιχεία της ομάδας.
  Οι δυνατότητες περιλαμβάνουν:
    - Αλλαγή ονόματος ή περιγραφής της ομάδας.
    - Προσθήκη ή αφαίρεση μελών (μόνο για τον Team Leader της ομάδας).
- **Διαγραφή ομάδας (Admin)**
  Οι διαχειριστές μπορούν να αφαιρούν ομάδες από το σύστημα όταν δεν χρειάζονται πλέον ή όταν η ομάδα έχει ολοκληρώσει το έργο της.
- **Προβολή ομάδων**
  Το API επιτρέπει στους χρήστες να βλέπουν τις ομάδες στις οποίες ανήκουν ή που διαχειρίζονται.
  Οι λειτουργίες προβολής περιλαμβάνουν:
    - Λίστα όλων των ομάδων που σχετίζονται με τον χρήστη.
    - Αναλυτικά στοιχεία κάθε ομάδας, όπως μέλη, περιγραφή και leader.
    - Ο Admin μπορεί να βλέπει όλες τις ομάδες.

Part 3: Task Management API

Το **Task Management API** ή πιο απλά **Task Service** είναι υπεύθυνο για **τη δημιουργία, οργάνωση και παρακολούθηση των εργασιών** (tasks) μέσα σε ομάδες.
Σκοπός του είναι να δίνει τη δυνατότητα στους χρήστες να διαχειρίζονται τις εργασίες τους, να παρακολουθούν την πρόοδο, να προσθέτουν σχόλια και να επισυνάπτουν αρχεία, δημιουργώντας ένα πλήρες σύστημα παρακολούθησης έργων.

Βασικές λειτουργίες του API

- **Δημιουργία εργασίας (Leader)**
  Οι αρχηγοί ομάδων μπορούν να δημιουργούν νέες εργασίες καθορίζοντας:
    - Τίτλο και περιγραφή εργασίας
    - Κατάσταση (status) – TODO (default)
    - Προτεραιότητα (π.χ. χαμηλή, μεσαία, υψηλή)
    - Ημερομηνία λήξης
    - Δημιουργό της εργασίας
    - Χρήστη στον οποίο έχει ανατεθεί η εργασία
- **Επεξεργασία εργασίας (Leader)**
  Το API επιτρέπει στους αρμόδιους χρήστες να τροποποιούν τα στοιχεία μιας εργασίας, όπως τίτλο, περιγραφή, assigned user, κατάσταση και due date.
- **Διαγραφή εργασίας (Leader)**
  Οι αρχηγοί ομάδων μπορούν να αφαιρούν εργασίες από το σύστημα όταν δεν χρειάζονται πλέον ή έχουν ολοκληρωθεί.
- **Προβολή εργασιών**
  Το API επιτρέπει στους χρήστες να βλέπουν λίστες εργασιών:
    - Ανά ομάδα ή ανά χρήστη, με δυνατότητα φιλτραρίσματος ανά κατάσταση (π.χ. μόνο TODO ή IN PROGRESS) ή προθεσμία ολοκλήρωσης
- **Αλλαγή κατάστασης εργασίας (Leader / Assigned User)**
  Οι χρήστες που έχουν την αρμοδιότητα μπορούν να ενημερώνουν την κατάσταση μιας εργασίας (π.χ. από TODO σε IN_PROGRESS ή DONE) για να παρακολουθείται η πρόοδος.
- **Σχόλια και επισυναπτόμενα αρχεία**
  Το API δίνει τη δυνατότητα στους χρήστες να προσθέτουν σχόλια σε κάθε εργασία (*προαιρετικά* να επισυνάπτουν και αρχεία που σχετίζονται με την εργασία).
- **Προβολή αναλυτικών στοιχείων εργασίας**
  Οι χρήστες μπορούν να βλέπουν όλες τις πληροφορίες μιας εργασίας της ομάδας που ανήκουν, συμπεριλαμβανομένων των σχολίων (και των επισυναπτόμενων αρχείων), ώστε να έχουν πλήρη εικόνα της εξέλιξης και του ιστορικού της εργασίας.

Part 4: Frontend / UI Service

Το **Frontend / UI Service** είναι η διεπαφή χρήστη που επιτρέπει την επικοινωνία με τα API.
Σκοπός του είναι να δίνει τη δυνατότητα στους χρήστες να βλέπουν και να διαχειρίζονται ομάδες και εργασίες ανάλογα με τον ρόλο τους, με **εύχρηστο και σαφή τρόπο**, σαν ένα απλοποιημένο Jira.

Ενδεικτικές Βασικές Σελίδες και Λειτουργίες

1. Login / Signup Page

- Είσοδος χρηστών με username / email και password.
- Δημιουργία νέου λογαριασμού (sign up) με username, email, όνομα, επώνυμο και password.
- Μήνυμα αν ο λογαριασμός χρειάζεται έγκριση από Admin.

2. Dashboard

- Επισκόπηση ομάδων και εργασιών που σχετίζονται με τον χρήστη.
- Προβολή βασικών στατιστικών:
    - Αριθμός ομάδων
    - Σύνολο εργασιών ανά κατάσταση (*TODO*, *IN_PROGRESS*, *DONE*)
    - Προσωπικές εργασίες
- Προαιρετικά: progress bars ή charts για οπτική αναπαράσταση της προόδου.

3. Teams Page

- Λίστα με όλες τις ομάδες που σχετίζονται με τον χρήστη.
- Προβολή βασικών στοιχείων: όνομα ομάδας, leader, αριθμός μελών.
- Λειτουργίες ανάλογα με ρόλο:
    - Admin/Leader: δημιουργία ομάδας, επεξεργασία στοιχείων, προσθήκη/αφαίρεση μελών.
- **Team Details / Tasks Page**:
    - Όταν ανοίγει μια ομάδα, εμφανίζεται η σελίδα της με **όλες τις εργασίες της ομάδας**.
    - Αναλυτικά στοιχεία ομάδας: όνομα, περιγραφή, leader, μέλη.
    - Λίστα / grid των εργασιών της ομάδας.
    - Προβολή βασικών πληροφοριών κάθε εργασίας (τίτλος, assigned user, κατάσταση, due date, priority).

4. Task Detailed Page

- Ξεχωριστή σελίδα για κάθε task.
- Όλες οι λεπτομέρειες της εργασίας: τίτλος, περιγραφή, assigned user, δημιουργός (createdBy), κατάσταση, due date, priority.
- Προβολή **σχολίων** και προαιρετικά **επισυναπτόμενων αρχείων**.
- Δυνατότητα επεξεργασίας ή αλλαγής κατάστασης ανάλογα με τον ρόλο του χρήστη.

5. My Tasks Page

- Λίστα με τις εργασίες που έχει αναλάβει ο χρήστης.
- Προβολή βασικών στοιχείων: τίτλος, κατάσταση, due date, priority.
- Δυνατότητα **άμεσης αλλαγής κατάστασης**.
- Πρόσβαση στην **Task Detailed Page** για περισσότερες λεπτομέρειες.

6. Admin Panel

- Λίστα με όλους τους χρήστες και τις ομάδες.
- Δυνατότητες: έγκριση χρηστών, αλλαγή ρόλων, απενεργοποίηση ή διαγραφή λογαριασμών, διαχείριση ομάδων.
- Κάθε στοιχείο ανοίγει σε **ξεχωριστή σελίδα λεπτομερειών** για αναλυτικές πληροφορίες και διαθέσιμες ενέργειες.

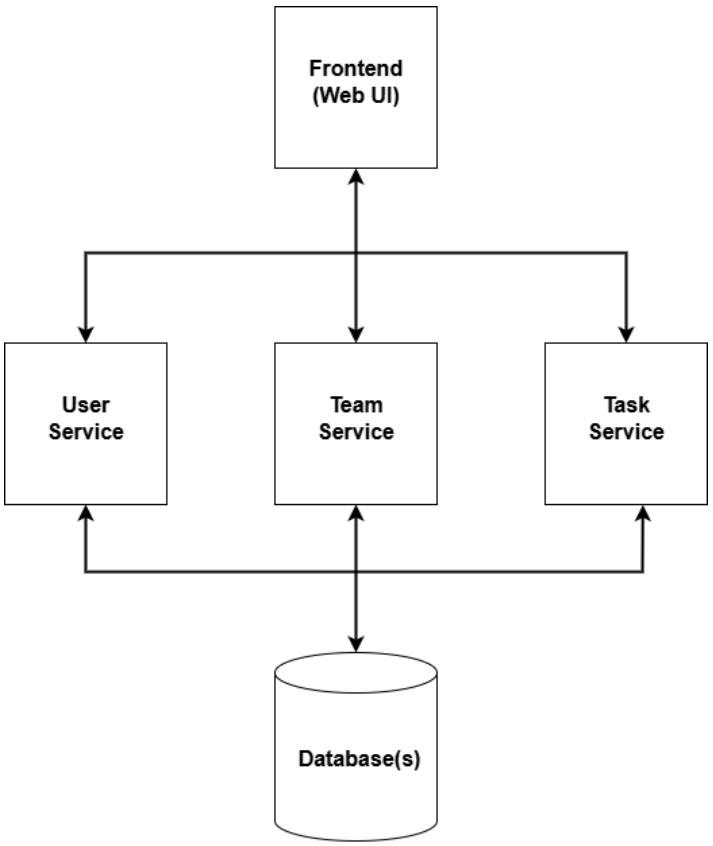
Σημείωση: Οι σελίδες που περιγράφονται είναι ενδεικτικές. Μπορείτε να προσθέσετε ή να τροποποιήσετε σελίδες, components και πλοήγηση, ανάλογα με τη σχεδίασή σας και τις επιλογές σας.

Part 5: Migration-deployment της εφαρμογής στο Google Cloud Platform (GCP)

Στο Google Cloud Platform, θα πρέπει να δημιουργήσετε ένα instance με λειτουργικό Ubuntu, να εγκαταστήσετε Docker (και docker-compose) σε αυτό και να σηκώσετε (deploy) εκεί την εφαρμογή. Θα χρειαστεί να μεταφέρετε στην εικονική μηχανή (VM) τα αρχεία του κώδικά σας και τα αντίστοιχα Dockerfiles, docker-compose, κλπ. Θα χρειαστεί επίσης να “ανοίξετε” (expose) έξω από το VM την κατάλληλη πόρτα (port) ή τις κατάλληλες πόρτες που αφορούν το Web User Interface (UI) ώστε να μας δώσετε πρόσβαση σε αυτό για να το δούμε και να το τεστάρουμε κατά τη διόρθωση-βαθμολόγηση των εργασιών. Προφανώς, στο GCP, η εφαρμογή σας θα πρέπει να είναι “UP and running” όταν σας ζητηθεί, π.χ. μετά από σχετική ανακοίνωση στο μάθημα.

Διάγραμμα Αρχιτεκτονικής (ενδεικτικό)

Το παρακάτω διάγραμμα αρχιτεκτονικής είναι ***ενδεικτικό*** (λειτουργεί βοηθητικά, όχι περιοριστικά) και απεικονίζει γενικά τις απαραίτητες υπηρεσίες της εφαρμογής που πρέπει να υλοποιηθούν και το πώς επικοινωνούν μεταξύ τους. Είστε ελεύθεροι να το τροποποιήσετε - επεκτείνετε όπως θέλετε και να “χτίσετε” την αρχιτεκτονική σας με βάση τη σχεδίαση και τις τεχνολογίες που σας εξυπηρετούν περισσότερο.



Προτεινόμενες Τεχνολογίες

Για την ανάπτυξη του συστήματος διαχείρισης έργων με microservices, σας προτείνουμε κάποιες τεχνολογίες για να έχετε οδηγό. **Δεν είναι υποχρεωτικό να τις χρησιμοποιήσετε όλες**, και μπορείτε να επιλέξετε **vanilla τεχνολογίες** (π.χ. pure Java ή PHP, JavaScript, HTML/CSS) αν θέλετε.

Backend / Microservices

- **Γλώσσες Προγραμματισμού:**
    - Java (π.χ. Spring Boot ή pure Java) ή PHP (π.χ. Laravel ή pure)
    - Python (π.χ. FastAPI, Django, Flask ή pure Python)
    - Node.js (π.χ. Express ή pure Node.js)
- **Βάσεις Δεδομένων:**
    - SQL: MySQL, PostgreSQL ή άλλες εναλλακτικές
    - NoSQL: MongoDB ή άλλες εναλλακτικές
- **API / Επικοινωνία μεταξύ services:**
    - REST API (μπορείτε να φτιάξετε και με pure HTTP requests)
- **Authentication & Authorisation:**
    - JWT Tokens για ασφαλή σύνδεση με λογαριασμό (login)
    - Μπορείτε να υλοποιήσετε και custom authentication χωρίς frameworks

Frontend / UI

- **Frameworks / Libraries:** Next.js, React, Angular ή άλλες εναλλακτικές
- **Vanilla τεχνολογίες:**
    - HTML, CSS, JavaScript χωρίς frameworks
    - DOM manipulation για διαχείριση components και UI
- **Charts / Visualisations:**
    - Chart.js, Recharts, ApexCharts ή απλά HTML/CSS/JS γραφήματα

Containers & Deployment

- Docker (κατά προτίμηση) ή Podman για containerization
- Docker/Podman Compose για τοπική ανάπτυξη πολλαπλών services

Σημείωση: Μπορείτε να χρησιμοποιήσετε όποιες τεχνολογίες θέλετε, ακόμα και καθαρές (vanilla) λύσεις, αρκεί να καλύπτονται οι βασικές απαιτήσεις:

- Microservices architecture
- Ασφαλής ταυτοποίηση (authentication)
- Διαχείριση ομάδων και εργασιών
- Εύχρηστο frontend User Interface (UI)

Παραδοτέο Άσκησης

Στο τέλος της άσκησης, θα πρέπει να παραδώσετε ένα ολοκληρωμένο σύστημα διαχείρισης έργων (Project Management System ή PMS) βασισμένο σε **microservices**. Το παραδοτέο σας πρέπει να περιλαμβάνει τα παρακάτω στοιχεία:

1. Backend / Microservices

- Όλα τα απαραίτητα services που αναπτύξατε:
    - **User Management API**
    - **Team Management API**
    - **Task Management API**
- Κάθε service πρέπει να παρέχει **REST API endpoints** για τις βασικές λειτουργίες που περιγράφονται στην άσκηση.
- Κάθε microservice θα έχει **δικό του API port** (π.χ. 8080, 8081, 8082, κλπ.).

2. Frontend / UI Service

- Ένα Frontend που επιτρέπει την επικοινωνία με τα API και υποστηρίζει διαφορετικούς ρόλους χρηστών. Το frontend μπορεί να επικοινωνεί **απευθείας με τα επιμέρους microservices μέσω REST calls**, ή, εναλλακτικά, να **μεσολαβεί ένα API Gateway** (π.χ. Nginx ή άλλο reverse proxy) για τη δρομολόγηση των αιτημάτων προς τα επιμέρους services.
- Προτεινόμενες σελίδες:
    - **Login / Signup Page**
    - **Dashboard**
    - **Teams Page** + Team Details / Tasks Page
    - **Tasks Page** + Task Detailed Page
    - **My Tasks Page**
    - **Admin Panel**
- Το UI πρέπει να επιτρέπει βασικές ενέργειες ανά ρόλο: προβολή, δημιουργία, επεξεργασία, ανάθεση εργασιών, αλλαγή status, σχόλια και προαιρετικά upload-attachment (μεταφόρτωση-επισύναψη) αρχείων.
- *Προαιρετικά* μπορείτε να προσθέσετε visualizations, charts ή progress bars.

3. Database / Storage

- Παραδώστε τα απαραίτητα αρχεία για τις βάσεις δεδομένων σας (π.χ. MySQL, MongoDB), όπου θα υπάρχουν ήδη κάποια sample data χρηστών, ομάδων, εργασιών, κλπ.

4. Docker / Containerization

- Αν χρησιμοποιήσατε Docker, παραδώστε:
    - Dockerfiles για κάθε service
    - Docker Compose αρχείο για τοπική ανάπτυξη (local deployment) όλων των services μαζί
- Αν δεν καταφέρατε να χρησιμοποιήσετε Docker, συμπεριλάβετε οπωσδήποτε **αναλυτικές οδηγίες εκτέλεσης των services τοπικά (locally)**.

5. Documentation

- Αναφορά της εργασίας με ό,τι κρίνετε απαραίτητο για την περιγραφή της εφαρμογής και των λειτουργιών της, τι υλοποιήσατε (και τι δεν υλοποιήσατε), κάτι ιδιαίτερο χρησιμοποιήσατε ή που κάνατε διαφορετικά, πώς δουλέψατε για το migration-deployment της εφαρμογής σας στο Google Cloud Platform και οδηγίες για το πού θα τη βρούμε (URL, port), κλπ.
- Σύντομο αρχείο README για όλο το project (documentation για εμάς και για άλλους developers), που περιλαμβάνει:
    - Πολύ σύντομη περιγραφή του project
    - Οδηγίες για εγκατάσταση και εκτέλεση Backend & Frontend (μπορεί να είναι απλά οι εντολές του Docker)
    - Οδηγίες για τη βάση δεδομένων
    - Οδηγίες χρήσης του UI

6. Προαιρετικά στοιχεία (BONUS)

- Προαιρετικά γραφήματα και visualizations στο Dashboard.
- Προαιρετική δυνατότητα επισύναψης αρχείων στις εργασίες (tasks) και στα σχόλιά τους από τους χρήστες.
- Προαιρετικές ασύγχρονες (event-driven) ειδοποιήσεις χρηστών (user notifications) στο User Interface για αλλαγές/deadlines σε tasks, με χρήση π.χ. Webhooks ή WebSockets ή Apache Kafka (ή άλλου μηχανισμού pub/sub), κλπ.

Σημαντικές Σημειώσεις

- Όλα τα services και το frontend πρέπει να είναι **λειτουργικά** και να επικοινωνούν σωστά μεταξύ τους.
- Έχετε την **ελευθερία επιλογής τεχνολογιών**, ακόμα και vanilla λύσεις, αρκεί να καλύπτονται οι βασικές λειτουργίες.
- Το παραδοτέο μπορεί να παραδοθεί ως **αρχεία project (code + Docker/DB configuration)** με αναλυτικό README και αναφορά εργασίας.

Βασικές Έννοιες και Τεχνολογίες

Για να ολοκληρώσετε την άσκηση, θα πρέπει πρώτα να κατανοήσετε μερικές βασικές έννοιες και τεχνολογίες που χρησιμοποιούνται σε σύγχρονα web συστήματα και microservices, ώστε να τις χρησιμοποιήσετε κατάλληλα κατά την υλοποίηση. Συνοπτικά, αναφέρονται παρακάτω:

API (Application Programming Interface)

- Το API είναι ένα σύνολο κανόνων και endpoints που επιτρέπουν σε δύο διαφορετικά συστήματα ή services να επικοινωνούν μεταξύ τους.
- Μέσω ενός API μπορείτε να στέλνετε δεδομένα, να δημιουργείτε, να ενημερώνετε ή να διαγράφετε πληροφορίες σε ένα backend service.
- Στην άσκηση, όλα τα services (User Management, Team Management, Task Management) θα εκθέτουν **API endpoints** για να τα χρησιμοποιεί το frontend.

REST (Representational State Transfer)

- REST είναι ένα στυλ αρχιτεκτονικής (architectural style) για τη δημιουργία σύγχρονων APIs.
- Χαρακτηριστικά REST API:
  - Χρήση HTTP μεθόδων: GET (ανάκτηση δεδομένων), POST (δημιουργία), PUT/PATCH (ενημέρωση), DELETE (διαγραφή)
  - Χρήση URLs για πρόσβαση σε resources (π.χ. /users, /teams, /tasks)
  - Επιστροφή δεδομένων συνήθως σε JSON format
- Στην άσκηση, τα microservices σας θα υλοποιηθούν ως HTTP **REST APIs** για απλή και ξεκάθαρη επικοινωνία.

Frontend / UI

- Το frontend είναι η διεπαφή χρήστη, δηλαδή αυτό που βλέπουν και χρησιμοποιούν οι χρήστες στο browser.
- Αλληλεπιδρά με τα API των backend services για να προβάλει δεδομένα και να δέχεται ενέργειες από τον χρήστη.
- Μπορεί να υλοποιηθεί με frameworks (React, Angular, Vue) ή με **vanilla HTML/CSS/JS**.

JWT (JSON Web Token)

- Χρησιμοποιείται για ασφαλή αυθεντικοποίηση/ταυτοποίηση (authentication).
- Όταν ένας χρήστης κάνει login, το backend δημιουργεί ένα token που περιέχει τα στοιχεία του χρήστη και τον ρόλο του.
- Το token στέλνεται στο frontend και χρησιμοποιείται για πρόσβαση στα protected endpoints των API.

Microservices

- Τα microservices είναι μια αρχιτεκτονική προσέγγιση όπου το σύστημα χωρίζεται σε μικρές, ανεξάρτητες υπηρεσίες.
- Κάθε service έχει συγκεκριμένη ευθύνη: π.χ. ένα για διαχείριση χρηστών, ένα για διαχείριση ομάδων, ένα για εργασίες.
- Πλεονεκτήματα:
    - Ανεξάρτητη ανάπτυξη και συντήρηση
    - Καλύτερη επεκτασιμότητα
    - Δυνατότητα να χρησιμοποιεί διαφορετικές τεχνολογίες ανά service

Docker/Podman

- Το Docker (ή Podman) σας επιτρέπει να “πακετάρετε” ένα service μαζί με όλες τις βιβλιοθήκες/εξαρτήσεις (libraries, dependencies, configuration, κλπ.) του σε ένα **container**, το οποίο μπορεί να τρέξει σε οποιοδήποτε σύστημα στο οποίο έχει εγκατασταθεί σωστά το Docker.
- Βασικές έννοιες που πρέπει να κατανοήσετε και να μάθετε να χρησιμοποιείτε: images, containers, volumes, networks, docker compose, κλπ.

- Για περιβάλλον Windows, προτείνουμε Docker Desktop ή Podman Desktop για προβολή, διαχείριση, έναρξη (start) και παύση (stop) Docker containers, διαχείριση Docker images και volumes, κλπ.
- Πλεονεκτήματα:
    - Απλή ανάπτυξη και δοκιμή των services
    - Σταθερό περιβάλλον για όλους τους χρήστες της ομάδας
    - Εύκολη διαχείριση πολλαπλών services μέσω του εργαλείου Docker Compose (docker-compose.yml) για configuration, deployment, κλπ.

Βάσεις Δεδομένων (Databases)

- Οι βάσεις δεδομένων είναι απαραίτητες για την αποθήκευση των δεδομένων του συστήματος και κατ' επέκταση τις λειτουργίες **CRUD**, δηλ. την εισαγωγή (Create), την ανάκτηση (Read), την επεξεργασία/ενημέρωση (Update) και διαγραφή (Delete) των δεδομένων.
- Τύποι βάσεων δεδομένων:
    - **SQL (Relational):** MySQL, PostgreSQL (κατάλληλες για δομημένα δεδομένα, όπως χρήστες και ρόλοι), θα μπορούσαν να χρησιμοποιηθούν για το User Management API (δική σας η επιλογή).
    - **NoSQL (Document/Key-Value):** MongoDB, Firebase (κατάλληλες για πιο ευέλικτα δεδομένα, όπως ομάδες και εργασίες), θα μπορούσαν να χρησιμοποιηθούν για το Team και το Task Management API (δική σας η επιλογή).

Good luck and…

Happy coding! 💻🚀