ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ

Στο μάθημα

PROJECT: ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΔΙΑΔΙΚΤΥΟΥ

ΟΜΑΔΑ 2

Αναγνωστόπουλος Άγγελος Νικόλαος
up1066593
Δροσιάδης Μιχαήλ
up1066594

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

10 ΙΟΥΝΙΟΥ 2022

Περιεχόμενα

1. Περιγραφή και απαιτήσεις προβλήματος

Σκοπός είναι η υλοποίηση ιστοσελίδας εύρεσης εργασίας. Για αυτό τον σκοπό θα χρειαστούν πληθώρα προγραμματιστικών τεχνικών καθώς οι απαιτήσεις είναι ανεβασμένες. Αρχικά, θα πρέπει να υπάρχουν χρήστες και μάλλιστα πολλών ειδών. Θα πρέπει να διατηρούνται sessions και να έχουμε τη δυνατότητα αποθήκευσης cookies των χρηστών σε μία γρήγορη βάση δεδομένων (redis). Οι χρήστες θα πρέπει να μπορούν να αναζητούν και να κάνουν αιτήσεις σε δουλειές, ενώ οι εταιρίες και οι αντιπρόσωποί τους θα πρέπει να μπορούν να αναρτούν νέες θέσεις εργασίας.

Αντιμετώπιση του προβλήματος

Μετά από πολλή σκέψη, σκεφτήκαμε ότι και για διδακτικούς σκοπούς, αλλά και για "σοβαρότητα" λύσης, θα είχαμε μία προσέγγιση με βάση το scalability, το availability και την ταχύτητα της ιστοσελίδας μας. Για αυτό οδηγηθήκαμε σε μία περίπλοκη αρχιτεκτονική η οποία περιέχει πολλούς διασυνδεδεμένους υπολογιστές και APIs, τα οποία διαχειρίζονται τα θέματα του backend.

Το github link μας: https://github.com/AngelosAnagnostopoulos/Amazing_Job_Finder

2. Σχεδίασμός

2.1. Σχεδιασμός αρχιτεκτονικής συστημάτων (Systems design)

Οι clients ζητάνε από έναν load balancer την ιστοσελίδα. Αυτός με τη σειρά του, ελέγχει την διαθεσιμότητα των application servers οι οποίοι τρέχουν την ιστοσελίδα. Κάθε ενέργεια του χρήστη αποτελέι και ένα αίτημα HTTP. Αυτό, ανάλογα με την φύση του, το διαχειρίζεται το κατάλληλο API. Παρακάτω υπάρχει το αναλυτικό διάγραμμα με το οποίο δουλέψαμε.

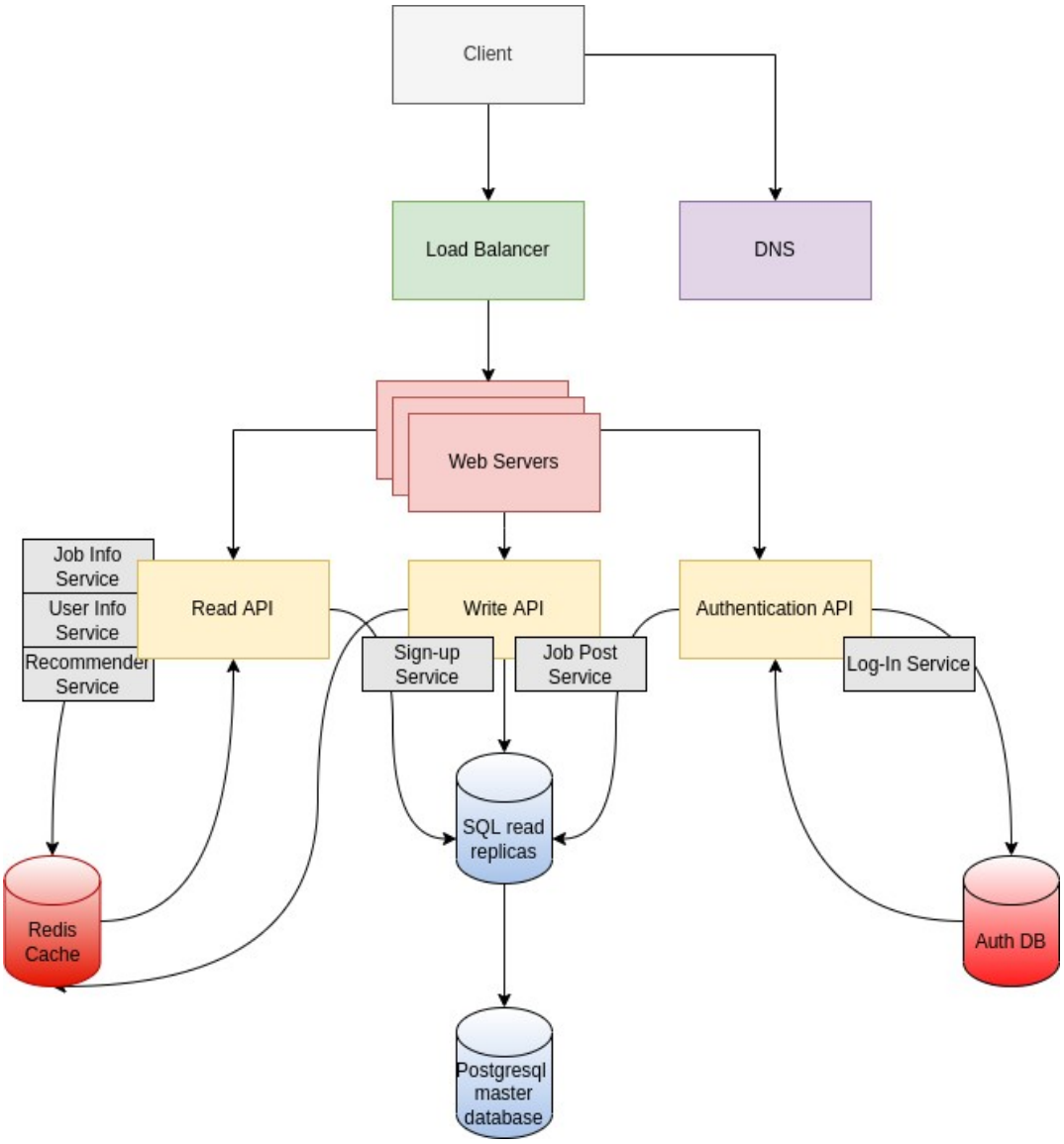
Το read μας δίνει τις πληροφορίες για τις υπάρχουσες δουλειές στην ΒΔ, φιλτράρωντας τες κατάληλα όταν του ζητηθεί.

Το write αφορά τις καταχωρήσεις νέων χρηστών και δουλειών.

Το authentication αφορά την είσοδο ενός υπάρχον χρήστη στην εφαρμογή.

Πράγματα που απεικονίζονται αλλά δεν υλοποιήθηκαν:

- Read replicas (Γράφουμε κατευθείαν στο slave database και δεν ασχοληθήκαμε με την αρχιτεκτονική αυτή παραπάνω).
- Redis cache (Αν και χρησιμοποιήθηκε τελικά redis για να κρατάει τα sessions, δεν την χρησιμοποιήσαμε για query caching).
- DNS (Το docker έχει internal DNS service και μας κάλυψε απόλυτα)



Σχήμα 1: Systems design diagram

2.2 Σχεδιασμός διεπαφής ιστοσελίδας (Frontend design)

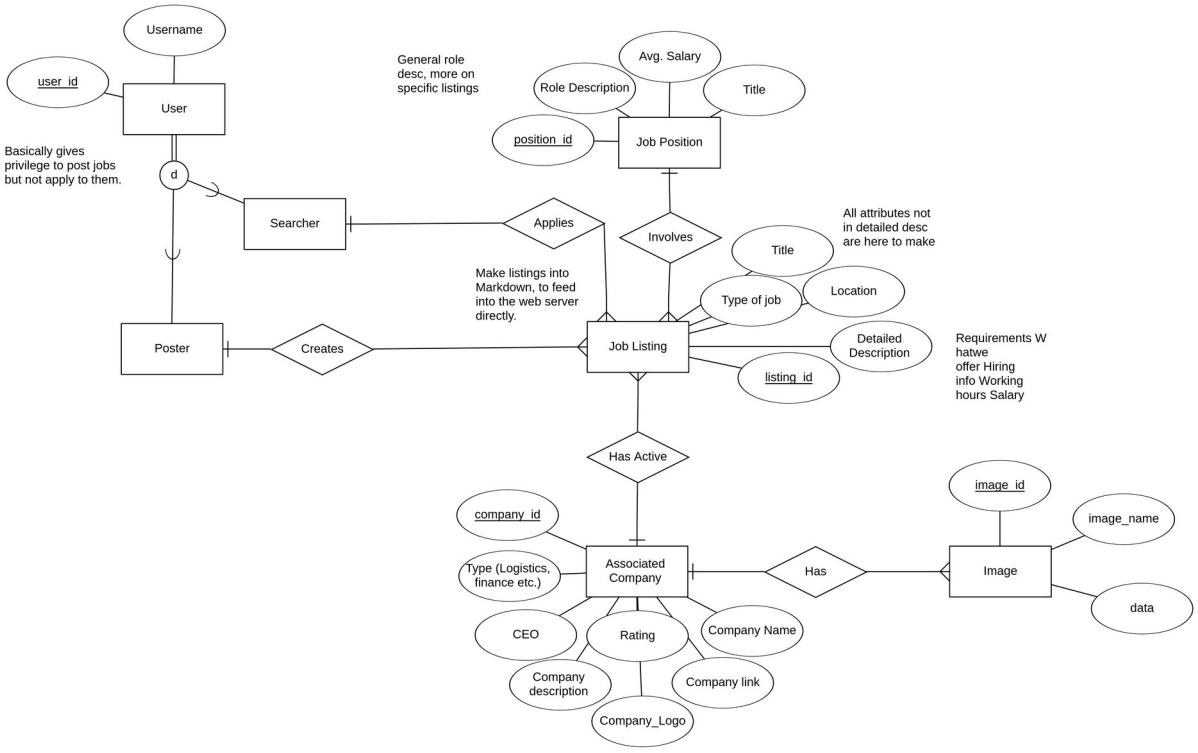
Ο σχεδιασμός του frontend έγινε από γραφίστα, η οποία μας έδωσε τα σχέδια και τα υλοποιήσαμε σε κώδικα HTML/CSS. Στόχος ήταν ένα φιλικό και μοντέρνο ντιζάιν, χωρίς να κουράζει τον χρήστη και με έμφαση στο να γίνονται τα περισσότερα πράγματα στη σελίδα χωρίς να αφήσει κάποιος το ποντίκι.

https://www.figma.com/file/mP4hV7A8BRnnHuC7A4CzTX/Job-finder-Snowflake?node-id=0%3A1

2.3 Σχεδιασμός βάσεων δεδομένων (Database design)

Οι βάσεις ήταν βασικό κομμάτι του πρότζεκτ, μιας και γύρω από αυτές έλαβε μορφή το backend μας. Η authentication database είναι ασήμαντη, αποτελείται από έναν πίνακα με username/password combination, με τον κωδικό να είναι κρυπτογραφημένος.

Η βασική βάση δεδομένων μας αποτελείται από πολλαπλούς πίνακες και καλύπτει χρήστες, εταιρίες, job listings και θέσεις εργασίας. Σκοπός ήταν να μπορούμε να φιλτράρουμε τις αναζητήσεις του χρήστη ανάλογα με τα ενδιαφέροντα του και τις εταιρίες.



Σχήμα 2: Database design diagram

2.4 Σχεδιασμός CI/CD pipeline

Φτιάξαμε μερικά απλά workflows στο github actions και τρέξαμε μερικά dummy tests, αναρτώντας τα αποτελέσματα στο codecov (χρησιμοποιώντας το action του). Αυτά θα χρησιμοποιηθούν για tests στις ΒΔ και τα APIs.

3. Τεχνολογίες

Frontend: HTML5, CSS3, Bootstrap5, Handlebars

APIs: NodeJS, Javascript

Servers/LoadBalancing: Nginx

Backend: Redis, PostgreSQL, MongoDB

Utils: Git/Github, Github Actions, Codecov ,Docker

4. Αξιοσημείωτα ζητήματα και παρατηρήσεις

Είναι γεγονός ότι με τον περιορισμένο χρόνο που είχαμε για να φτιάξουμε το backend αργήσαμε κάποιες μέρες παραπάνω. Παρόλα αυτά, αυτό έγινε διότι υπήρξαν προκλήσεις με βιβλιοθήκες διασύνδεσης στις βάσεις μας (η official βιβλιοθήκη της redis στην javascript είχε bug και η λύση τους ήταν “Χρησιμοποιήστε το legacy mode της περασμένης έκδοσης”), με τον load balancer και την αποθήκευση των sessions, με τα πολλαπλά containers κ.α.

Τα animations γενικά ήταν κάπως χρονοβόρα και οι μικροδιορθώσεις στην CSS μας κράτησαν κάπως πίσω.

Η javascript ήταν αρκετά ταλαιπωρία και μερικές φορές αρνούνταν να συνεργαστεί με την bootstrap.

Και άλλα πολλά…

5. Μελλοντικές επεκτάσεις

1. Redirects στο sign-up/log-in text στην βάση των αντίστοιχων modals για χρήστες που δεν έχουν ή έχουν ήδη λογαριασμό.
2. Υλοποίηση εταιριών και χρηστών που δουλεύουν για αυτές.
3. Query caching σε μία redis database.
4. Unit tests για τις βάσεις δεδομένων και τα APIs.