**Project Προγραμματισμός και Συστήματα στον Παγκόσμιο Ιστό**

**Ονοματεπώνυμο/ΑΜ:**

Περικλής Κοροντζής / 1072563

Γρηγόριος Γεώργιος Καπαδούκας / 1072484

Ηλίας Σερταρίδης / 1072480

**Έτος:** 2022 – 2023

**Εξάμηνο:** 7ο

**Email:** up1072563@upnet.gr

up1072484@upnet.gr

up1072480@upnet.gr

**0] Stack Εφαρμογής**

Για τη δημιουργία της εφαρμογής χρησιμοποιήσαμε Apache, PHP, MySQL και JavaScript με επιπλέον χρήση των βιβλιοθηκών Bootstrap, JQuery, Leaflet.js, Chart.js και DataTables. Η αρχιτεκτονική της εφαρμογής μας αποτελείται επίσης από Docker containers.

**A] Σχεδιασμός Βάσης**

Το ER Diagram της βάσης που υλοποιήσαμε για την εκπόνηση του project παρουσιάζεται παρακάτω:

A diagram of a company

Description automatically generated

Με βάση το παραπάνω διάγραμμα και τους κανόνες σχεδιασμού σχεσιακού μοντέλου που διδαχθήκαμε στο μάθημα των βάσεων δεδομένων, δημιουργήσαμε τους τελικούς πίνακες που απαρτίζουν τη βάση του συστήματός μας και βρίσκονται στο αρχείο create.sql με relative path:

e-deals interactive website/SQL/create.sql .

**Επεξήγηση:**

Η ιστοσελίδα που μας ανατέθηκε να δημιουργήσουμε είναι μια πλατφόρμα μεταφόρτωσης και αξιολόγησης προσφορών για προϊόντα τοπικών super market και παντοπωλείων της Πάτρας. Συνεπώς στην βάση μας θα πρέπει να αποθηκεύουμε τα προϊόντα, τις τιμές τους, καθώς και τις προσφορές για αυτά τα προϊόντα. Παρά το γεγονός ότι μας δόθηκαν τα δεδομένα των καταστημάτων σε έτοιμο GeoJSON αρχείο, επιλέξαμε να αποθηκεύσουμε τις βασικές πληροφορίες για κάθε κατάστημα στην βάση μας. Ο λόγος για αυτή την απόφαση είναι ότι μας έκανε ευκολότερη την δημιουργία query για τη συγκέντρωση των πληροφοριών που ζητούσε ο χρήστης από τη βάση, αλλά και τη συσχέτιση των καταστημάτων με τις προσφορές τους.

Εδώ θα θέλαμε να αναφέρουμε πως η MySQL δεν υποστηρίζει αποθήκευση GeoJSON αρχείου αλλά υποστηρίζει αποθήκευση συντεταγμένων με data type point, και συνεπώς οι πλειάδες του πίνακα Store είναι store\_id, name, coordinates, και store type. Για την εμφάνιση, όμως, των καταστημάτων στον χάρτη (γίνεται χρήση της βιβλιοθήκης Leaflet.js), τα δεδομένα μας πρέπει να είναι σε μορφή GeoJSON. Για το λόγο αυτό υλοποιήσαμε το API μας να επιστρέφει αυτόματα GeoJSON arrays για ερωτήματα σχετικά με τα stores, τα οποία φορτώνονται στο Leaflet.

Πέρα των προϊόντων, των προσφορών και των καταστημάτων, θέλουμε να αποθηκεύουμε στη βάση και πληροφορίες για τους χρήστες της ιστοσελίδας, είτε είναι απλοί χρήστες, είτε είναι admins. Όλες οι πληροφορίες που επιθυμούμε να γνωρίζουμε για τον κάθε χρήστη και τη δραστηριότητά του παρουσιάζονται στο ER Diagram ως πλειάδες. Οι αξιολογήσεις που υποβάλλει ο κάθε χρήστης για κάποια προσφορά παρουσιάζονται στο προαναφερθέν διάγραμμα ως συσχέτιση με ένα χαρακτηριστικό (rating), ενώ στην βάση μας αποτελούν ξεχωριστό πίνακα. Επιπλέον, με βάση την εκφώνηση του project, κάθε προϊόν ανήκει σε μία υποκατηγορία και κάθε υποκατηγορία σε μία βασική κατηγορία προϊόντων. Οι δυο αυτές οντότητες που συσχετίζονται μεταξύ τους με 1-Ν συσχέτιση αντιστοιχούν σε δύο πίνακες στη βάση μας.

Τέλος, να σημειωθεί πως επιλέξαμε να υποδείξουμε τον πίνακα offer ως ασθενή οντότητα διότι αν δεν υπάρχουν δεδομένα σε έναν από τους πίνακες Item, Store, και User τότε δεν είναι δυνατόν να τοποθετηθεί κάποια προσφορά. Παρόλα αυτά θέσαμε ως primary key του Offer το offer\_id καθώς είναι μοναδικό αναγνωριστικό και το προτιμούμε ως κλειδί (σε σύγκριση με τη χρήση των foreign keys ως primary key) για λόγους απλότητας.

**Β] Ρυθμίσεις και Παραμετροποιήσεις**

Παρακάτω αναφέρουμε και επεξηγούμε τις ρυθμίσεις που κάναμε στον server μας ώστε να αξιοποιήσουμε κατάλληλα τις δυνατότητες των κρυφών μνημών. Οι ρυθμίσεις αυτές βρίσκονται στο αρχέιο 000-default.conf με relative path: e-deals interactive website/config/sites-available/000-default.conf .

Επιλέξαμε να χρησιμοποιήσουμε το 000-default.conf έναντι του .htaccess καθώς οι ρυθμίσεις του 000-default.conf φορτώνονται μία μόνο φορά κατά την εκκίνηση του server, ενώ το .htaccess αρχείο διαβάζεται για κάθε αίτημα χρήστη, κάτι το οποίο ενδέχεται να επιβραδύνει τον server.

* **CacheQuickHandler off**: απενεργοποιούμε το quick handler feature. Όταν ο χρήστης κάνει ένα αίτημα στον server, το feature αυτό επιτρέπει στον server να ψάξει στην κρυφή μνήμη προτού ολοκληρώσει την επεξεργασία του αιτήματος, ώστε να δώσει την απάντηση γρηγορότερα και να μην καταναλωθούν υπολογιστικοί πόροι. Ωστόσο στην ιστοσελίδα μας ασχολούμαστε με δυναμικό περιεχόμενο, συνεπώς δεν μπορούμε να βασιστούμε στην τοπική cache του κάθε χρήστη για να δώσουμε μια ακριβή απάντηση στο αίτημά του.
* **CacheLock on:** επιτρέπουμε τη χρήση των cache locks καθώς επιθυμούμε να υπάρχει αμοιβαίος αποκλεισμός μεταξύ processing threads που επιθυμούν να επεξεργαστούν το ίδιο περιεχόμενο της cache.
* **CacheLockPath /tmp/mod\_cache-lock:** ορίζουμε το directory στο οποίο θα αποθηκεύονται τα cache lock αρχεία. Τα αρχεία αυτά μας δείχνουν κάθε στιγμή ποια δεδομένα της cache είναι δεσμευμένα.
* **CacheLockMaxAge 5:** θέτουμε τη μέγιστη ηλικία, δηλαδή χρόνο, που ένα cache lock μπορεί να δεσμεύσει κάποιο περιεχόμενο της cache. Αν ο χρόνος αυτός παρέλθει και το cache lock δεν έχει αποδεσμεύσει το περιεχόμενο, τότε το περιεχόμενο απελευθερώνεται αυτόματα.
* **CacheIgnoreHeaders Set-Cookie:** καθώς η ιστοσελίδα μας λειτουργεί με sessions, τα session cookies μπορεί να χρησιμοποιηθούν για session management. Συνεπώς, όταν κάποιος χρήστης κάνει ένα αίτημα στον server, θέλουμε το Set-Cookie header να αγνοείται και το αίτημα να αντιμετωπίζεται σαν το header να μην υπάρχει, αποφεύγοντα έτσι θέματα ασφάλειας προσωπικών δεδομένων των χρηστών.
* **CacheEnable disk:** ενεργοποιούμε το caching στον server μας και επιλέγουμε τα cached responses να αποθηκεύονται στο disk. Το disk είναι το τοπικό storage device του server μας, δηλαδή το hard drive ή solid-state drive (SSD) του server μας. Η αποθήκευση των cached responses σε disk επιτρέπει την μακροχρόνια αποθήκευση δεδομένων, τη διαχείριση μεγάλου όγκου δεδομένων, αλλά και τη γρήγορη ανάκτηση των δεδομένων από το disk.
* **CacheHeader on:** ενεργοποιούμε το caching των response headers, οι οποίοι μας επιτρέπουν να γνωρίζουμε αν και πόσο ένα response μπορεί να γίνει cached χωρίς να διαβάσουμε το body του request.
* **CacheDefaultExpire 800:** θέτουμε ως προκαθορισμένο χρόνο λήξης εγκυρότητας μίας cached πληροφορίας τα 800 seconds ή 16 λεπτά. Το CacheDefaultExpire αφορά όλο το caching module του server, και καθορίζει το χρονικό διάστημα κατά το οποίο μια cached πληροφορία θεωρείται έγκυρη, όταν δεν έχουν προσδιοριστεί άλλες caching ρυθμίσεις αναφορικά τη λήξη εγκυρότητας ενός cached response.
* **CacheMaxExpire 64000:** θέτουμε ως μέγιστο χρόνο λήξης εγκυρότητας μιας cached πληροφορίας τα 64000 seconds.
* **CacheIgnoreNoLastMod On:** ενεργοποιούμε το caching των responses που δεν περιέχουν το Last-Modified header.
* **ExpiresActive on:** ενεργοποιούμε τη χρήση του Expires header, ώστε να μπορούμε να ορίσουμε Expires header στα http responses. Εφόσον η εντολή αυτή περικλείεται από <Location /> </Location> clauses, το header αυτό τοποθετείται σε όλα τα responses που ξεκινάνε από το root του server.
* **ExpiresDefault A300:** ορίζουμε το Expires header για τα http responses να είναι στα 300 seconds ή 5 λεπτά. Αυτό σημαίνει πως όλα τα responses με Expires header θα γίνονται cached για 5 λεπτά, εκτός αν ζητηθεί ανανέωση από το χρήστη.

Στο τέλος του 000-default.conf ενεργοποιήσαμε και το rewrite module το οποίο κάνει redirect όλα τα requests του χρήστη (με εξαίρεση τα requests προς συγκεκριμένους τύπους αρχείων, όπως εικόνες) σε ένα application level router που χρησιμοποιούμε, συγκεκριμένα το PHPRouter (phprouter.com), μέσω του οποίου δρομολογούμε με βάση του URL που ζητείται στα αντίστοιχα PHP αρχεία, έτσι αποκρύβοντας από τον τελικό χρήστη το εσωτερικό file system του server (συγκεκριμένα το φάκελο /var/www/html του Apache server) και λαμβάνοντας τη δυνατότητα για advanced routing δυνατότητες, όπως dynamic GET routes.

Πέρα των παραπάνω configurations, προσθέσαμε και μερικές εντολές για θέσμιο headers σε PHP αρχεία, και πιο συγκεκριμένα την εντολή header("Cache-Control: no-store"); καθώς δεν θέλουμε να γίνονται cached τα responses που μας δίνονται από αυτά τα PHP αρχεία.

Παραδείγματος χάρη, παρακάτω παρουσιάζουμε ένα screenshot από τα developer tools του chrome όπου ένας χρήστης έχει επιλέξει να δει τις προσφορές ενός καταστήματος στη σελίδα review:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

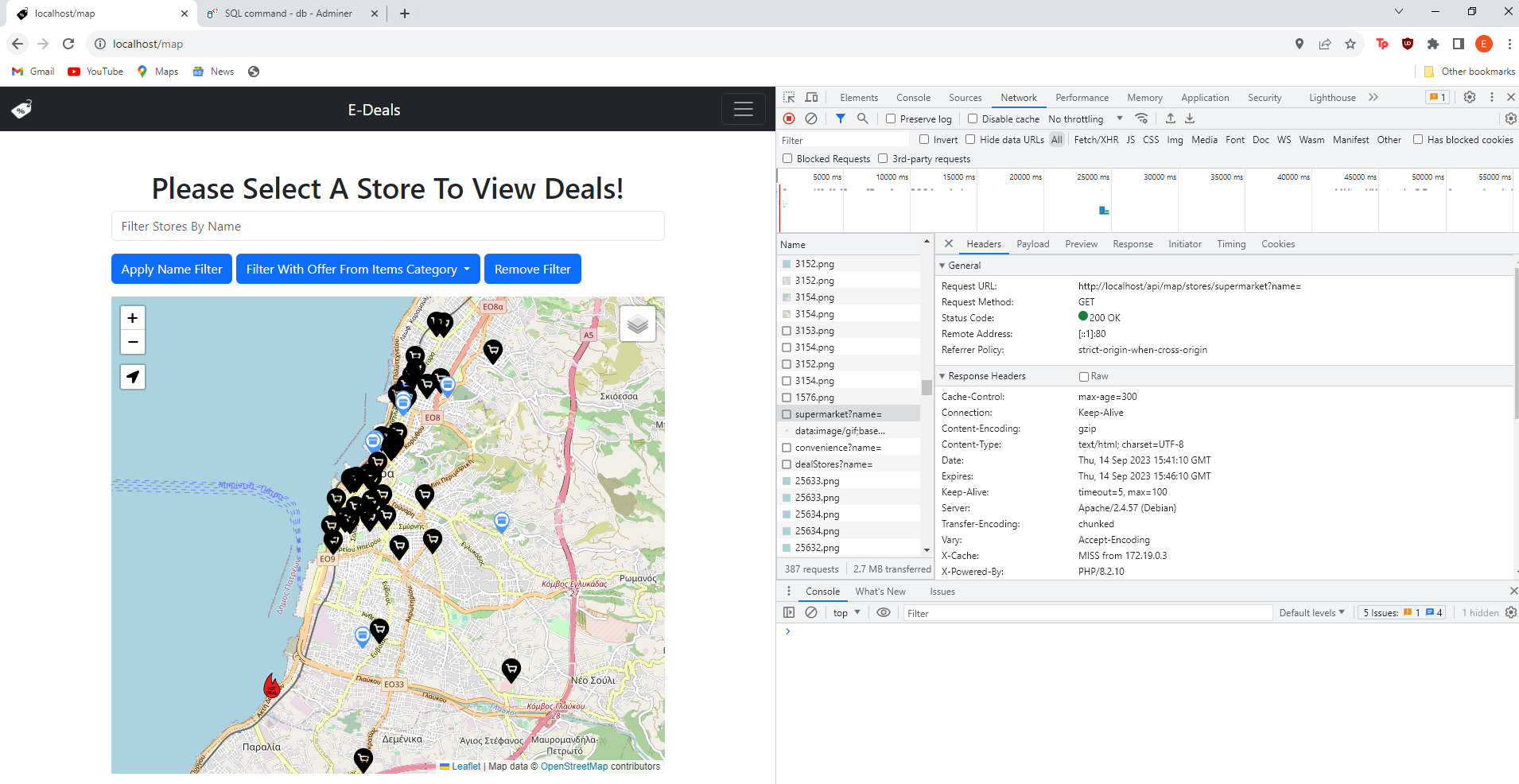
Όταν ο χρήστης μεταβαίνει στη σελίδα review, καλείται η συνάρτηση displayOffers η οποία με τη σειρά της εκτελεί ένα AJAX query μέσω jQuery ώστε να εμφανίσουμε στον browser τις προσφορές που έχουν υποβληθεί για το συγκεκριμένο κατάστημα. Επειδή οι πληροφορίες που εμφανίζονται στον χρήστη από αύτη του την αναζήτηση ανανεώνονται συχνά, επιλέξαμε να μην αποθηκεύουμε τα περιεχόμενα αυτής της αναζήτησης στην cache του browser του χρήστη ώστε κάθε φορά που θα ανανεώνει τη σελίδα του, να λαμβάνει την τελευταία έκδοση των δεδομένων που του παρουσιάζονται.

Αντιλαμβανόμαστε πως αν θέλαμε να χρησιμοποιήσουμε την εφαρμογή μας στο πραγματικό κόσμο, η ιδανική λύση θα ήταν να χρησιμοποιήσουμε ένα μικρότερο max-age σε συνδυασμό με caching σε αυτή τη περίπτωση, αλλά επιλέξαμε το no-store για να μπορούμε να δείξουμε την ανανέωση της σελίδας μας σε πραγματικό χρόνο.

Οπότε το max-age=300 δεν έχει κάποια σημασία εδώ καθώς δεν η σελίδα αυτή δεν αποθηκεύεται στην κρυφή μνήμη.

Παρόλα αυτά, σε άλλες περιπτώσεις όπως στη login σελίδα σε καμία περίπτωση δεν θέλουμε να γίνεται caching του response, οπότε εκεί θα χρησιμοποιούσαμε πάντα no store.

Ωστόσο, σε άλλες σελίδες όπως στη σελίδα του χάρτη, κάνουμε cache τα responses από τα queries που μας επιστρέφουν τα καταστήματα καθώς αποτελούν δεδομένα που δεν μεταβάλλονται συχνά (και σε αυτή την περίπτωση έχουμε max-age=300):



A screenshot of a computer

Description automatically generated

**Γ] Τρόπος εκτέλεσης του project**

Θέλοντας να είμαστε σίγουροι ότι ο κώδικας της ιστοσελίδας μας θα έτρεχε το ίδιο σε κάθε συσκευή στην οποία δούλευε το κάθε μέλος της ομάδας, αλλά και σε τυχόν server στη περίπτωση που θέλαμε να τον κάνουμε deploy, αποφασίσαμε να στήσουμε την σελίδα σε περιβάλλον docker.

Προκειμένου να μπορέσετε να εκτελέσετε τον κώδικά μας, θα πρέπει να τρέξετε στον τοπικό σας υπολογιστή το docker, να ανοίξετε το terminal στον φάκελο του project και να εκτελέσετε την εντολή docker compose up -v. Η εντολή αυτή θα δημιουργήσει 3 containers, το PHP container μας που περιέχει και τον Apache server, το MySQL container και το db-admin container που χρησιμοποιούμε για να κάνουμε manual management του περιεχομένου της βάσης σε GUI environment.

Πιο συγκεκριμένα, το PHP container τρέχει στη θύρα 80 του host machine, το MySQL container στις θύρες 3306 και 33060 και το db-admin container στη θύρα 8080.

Γράφοντας localhost στον περιηγητή της προτίμησής σας θα μπορέσετε να δείτε την δουλειά μας. Όταν ανοίξετε για πρώτη φορά την σελίδα η βάση θα κάνει αυτόματα initialize τρέχοντας τα αρχεία στο φάκελο SQL (relative path: e-deals interactive website/SQL). Ωστόσο, έχουμε συμπεριλάβει και δύο αρχεία με ονόματα dump.sql και import\_optional\_data.sql τα οποία βρίσκονται στον φάκελο e-deals interactive website/Database export, τα οποία μπορείτε να τρέξετε και να βάλετε στην βάση όλα τα δεδομένα που χρειάζεστε για να κάνετε δοκιμές για τη σωστή λειτουργία της ιστοσελίδας μας.

Το import\_optional\_data.sql περιέχει δεδομένα χρήσης της σελίδας, οπότε η εκτέλεση του θα κάνει populate την πλατφόρμα με offers, reviews, users, items, stores και οτιδήποτε άλλο χρειάζεται για να παρουσιαστούν οι πλήρεις λειτουργικότητες της εφαρμογής. Το dump.sql περιέχει πλήρη export της βάσης με το περιεχόμενο όλων των άλλων SQL αρχείων, δηλαδή των αρχείων του φακέλου SQL και του import\_optional+\_data.sql.

Για να τρέξετε τα αρχεία θα γράψετε στον browser localhost:8080, το οποίο θα σας μεταβιβάσει στον db Adminer (service του db-admin). Για να κάνετε login στη βάση μας, το όνομα και ο κωδικός είναι root ενώ το όνομα της βάσης είναι edeals. Στο παράθυρο που θα σας βγάλει θα πατήσετε SQL command και στο κεντρικό πλαίσιο θα γράψετε execute SQL και μετά copy paste τον κώδικα του κάθε αρχείου (πρώτα το dump και μετά το optional\_data). Εναλλακτικά μπορείτε να κάνετε χρήση της λειτουργίας import και να ανεβάσετε το αρχείο SQL που θέλετε να εκτελέσετε.

Περισσότερες πληροφορίες μέσω των configurations των containers βρίσκονται στα αρχεία docker-compose.yml και στο Dockerfile. Για να κλείσετε το container γράφετε στο terminal docker compose down -d.