**Ασφαλής προγραμματισμός**

Εργασία 2019-2020

Αναγνωστάκης Γεώργιος 319003

Βαγγελάτος Παναγιώτης 319005

Κανελλόπουλος Χρήστος 19016

Σακελλαρίου Αποστόλης 19042

****

Περιεχόμενα

[Εισαγωγή 3](#_Toc51264312)

[Spring Security 3](#_Toc51264313)

[Credentials 4](#_Toc51264314)

[Crawling 5](#_Toc51264315)

[Injection prevention 8](#_Toc51264316)

[Session management 10](#_Toc51264317)

[Server hardening 13](#_Toc51264318)

[Code auditing 15](#_Toc51264319)

[Παρουσίαση εφαρμογής 16](#_Toc51264320)

# Εισαγωγή

Στην διαδικτυακή πλατφόρμα που υλοποιήθηκε δίνουμε την δυνατότητα ένας χρήστης να εγγραφεί και είτε να πουλήσει (seller) είτε να αγοράσει (buyer) αντικείμενα. Έχει γίνει χρήση του Spring Security καθώς και άλλων εργαλείων ώστε να προστατεύσουμε στο μέγιστο εφικτό βαθμό την εφαρμογή μας. Παρακάτω θα δούμε ποια εργαλεία χρησιμοποιήθηκαν προκειμένου να βρούμε ή να προστατευτούμε από θέματα που αφορούν την ασφάλεια της εφαρμογής μας.

# Spring Security

Το Spring Security είναι ένα framework που επιδιώκει να εξασφαλίσει την ασφάλεια των διαδικτυακών εφαρμογών βασισμένων στο Spring Framework. Το Spring Security παρέχει υπηρεσίες ελέγχου ταυτότητας και πρόσβασης, ενώ παράλληλα δίνει τη δυνατότητα στον προγραμματιστή να επιλέξει τη μέθοδο κρυπτογράφησης των κωδικών χωρίς να γνωρίζει το σύνολο των ρυθμίσεων που απαιτούνται στο παρασκήνιο για την ορθή λειτουργία του framework. Επίσης, παρέχει προστασία από ορισμένους τύπους επιθέσεων όπως Enumeration attacks, SQL Injection, Brute Force Authentication, Session Hijacking και CSRF επιθέσεις.

Η αρχιτεκτονική του Spring Security είναι σχεδιασμένη έτσι ώστε να διαχωρίζει τις υπηρεσίες ελέγχου ταυτότητας (Authentication) από τις υπηρεσίες ελέγχου πρόσβασης (Authorization). Η βασική στρατηγική που χρησιμοποιείται για τον έλεγχο ταυτότητας είναι ο AuthenticationManager όπου χαρακτηρίζει ένα αίτημα ταυτοποίησης ως έγκυρο, άκυρο ή αβέβαιο εφόσον δε μπορεί να προκύψει ασφαλές συμπέρασμα ότι η προσπάθεια ταυτοποίησης γίνεται από τον πραγματικό χρήστη. Ο έλεγχος ταυτότητας αφορά το Login.

Ο έλεγχος πρόσβασης γίνεται μέσω του μηχανισμού AccessDecisionManager που σε κάθε προσπάθεια του χρήστη να αποκτήσει πρόσβαση σε μια συγκεκριμένη πηγή ή μέθοδο, επιστρέφει με βάση τις προδιαγραφές που έχουν δοθεί ένα αλφαριθμητικό, που αφορά τους κανόνες πρόσβασης στην πηγή. Για να εξασφαλιστεί ο έλεγχος πρόσβασης, αποδίδεται σε κάθε κατηγορία χρηστών ένας ρόλος, ο οποίος παρέχει συγκεκριμένα δικαιώματα πρόσβασης.

Εικόνα που περιέχει στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα 1: Οι βασικές λειτουργίες του Spring Security

# Credentials

Ευπάθειες Eλέγχου Πρόσβασης

Η διασφάλιση της ιδιωτικότητας των προσωπικών δεδομένων που διαχειρίζεται μια διαδικτυακή εφαρμογή επιτυγχάνεται μέσω του περιορισμού του αριθμού των χρηστών που έχουν το δικαίωμα πρόσβασης στα δεδομένα αυτά. Ο πιο συνήθης τρόπος υλοποίησης του παραπάνω είναι η παροχή διαπιστευτηρίων στους ενδιαφερόμενους χρήστες (π.χ. username, password).

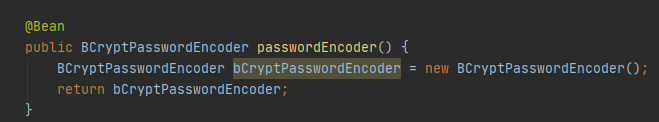
Η έλλειψη ή η ανεπαρκής υλοποίηση του ελέγχου των διαπιστευτηρίων αυτών μπορεί να οδηγήσει σε παραβίαση της ιδιωτικότητας των δεδομένων, δηλαδή τη δυνατότητα πρόσβασης σε αυτά από μη – διαπιστευμένους χρήστες. Τις συγκεκριμένες ευπάθειες τις εκμεταλλεύονται οι επιθέσεις Authentication Bypass και Authorization Bypass. Oι επιθέσεις αυτές συνήθως υλοποιούνται με αλλοίωση του URL ώστε να παρακαμθεί το στάδιο της εγγραφής (sign in) ή της σύνδεσης όπου αυτό (log in) απαιτείται.

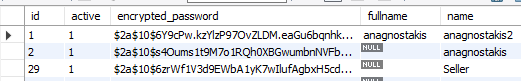
Δηλαδή το πρόβλημα που έχουμε να αντιμετωπίσουμε είναι η λανθασμένη υλοποίηση λειτουργιών που σχετίζονται με τις συνόδους (sessions) που έχουν ως αποτέλεσμα τη δυνατότητα υποκλοπής συνθηματικών, κλειδιών ή άλλων στοιχείων.

Αντιμετώπιση του broken authentication

Η κυριότερη σύσταση για την αποτροπή της ευπάθειας αυτής είναι η χρήση ισχυρών εργαλείων αυθεντικοποίησης του session. Στην εργασία αυτή για την αντιμετώπιση του προβλήματος αυτού χρησιμοποιήθηκε ο αλγόριθμος Bcrypt που είναι σχεδιασμένος για την προστασία των κωδικών πρόσβασης. Λανθασμένη υλοποίηση λειτουργιών που σχετίζονται με τις συνόδους που έχουν ως αποτέλεσμα τη δυνατότητα υποκλοπής συνθηματικών, κλειδιών ή άλλων στοιχείων συνόδων.

Ο αλγόριθμος Bcrypt εφαρμόζει κατακερματισμό κωδικού πρόσβασης με βάση την κρυπτογράφηση Blowfish που παρουσιάστηκε στο USENIX το 1999. Ο συγκεκριμένος αλγόριθμος είναι ο προεπιλεγμένος για το OpenBSD και άλλα συστήματα όπως και ορισμένων διανομών Linux.





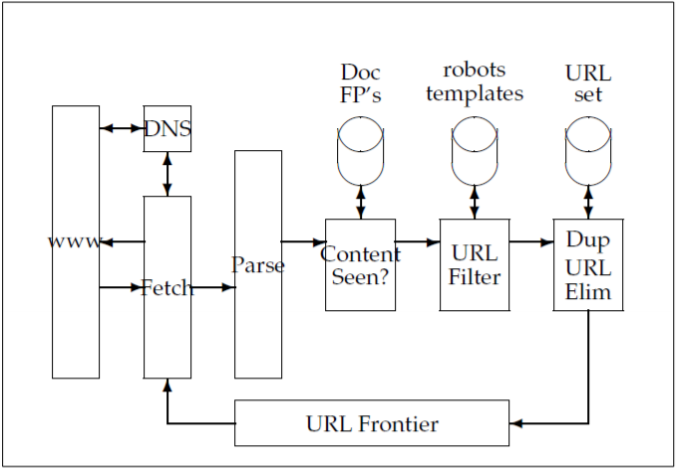
# Crawling

Η βασική λειτουργία οποιουδήποτε Crawler υπερκειμένου (είτε για τον ιστό, είτε για μία άλλη συλλογή εγγράφων υπερκειμένου) έχει ως εξής:

Ο ανιχνευτής αρχίζει με μία ή περισσότερες διευθύνσεις URL που αποτελούν ένα σύνολο σπόρων (seed set) . Επιλέγει μια διεύθυνση URL από αυτό το σύνολο σπόρων, στη συνέχεια ανακτά την ιστοσελίδα σε αυτήν τη διεύθυνση URL. Στη συνέχεια, η ανακτημένη σελίδα αναλύεται (parsed) για την εξαγωγή τόσο του κείμενου όσο και των συνδέσμων από τη σελίδα (καθένας από τους οποίους δείχνει σε άλλη διεύθυνση URL). Το εξαγόμενο κείμενο τροφοδοτείται σε ένα ευρετηριαστή κειμένου (text indexer). Στη συνέχεια, οι σύνδεσμοι που έχουν ανακαλυφθεί (διευθύνσεις URL) προστίθενται σε μια διεύθυνση URL συνόρου, το οποίο ανά πάσα στιγμή αποτελείται από διευθύνσεις URL των οποίων οι αντίστοιχες σελίδες πρέπει ακόμη να ανακτηθούν από τον crawler. Αρχικά, το URL συνόρου περιέχει το σύνολο σπόρου· καθώς ανακτώνται οι σελίδες, οι αντίστοιχες διευθύνσεις URL διαγράφονται από τη διεύθυνση URL σύνορο. Η όλη διαδικασία μπορεί να θεωρηθεί ως διάσχιση του γραφήματος του ιστού. Σε συνεχή ανίχνευση, προστίθεται η διεύθυνση URL μιας σελίδας που έχει ανακτηθεί ξανά στο σύνορο για να επανακτηθεί στο μέλλον.

Αυτή η φαινομενικά απλή αναδρομική διάσχιση του διαγράμματος ιστού περιπλέκεται από τις απαιτήσεις που απαιτούνται για ένα πρακτικό σύστημα ανίχνευσης: ο ανιχνευτής πρέπει να είναι κατανεμημένος, κλιμακωτός, αποτελεσματικός, ευγενικός, εύρωστος και επεκτάσιμος, ενώ να παράλληλα θα μπορεί να ανακτά σελίδες υψηλής ποιότητας.

Αυτή η αρχιτεκτονική αντιστοιχεί στο παρακάτω διάγραμμα που αποτελεί μία πιο λεπτομερή εκδοχή της απλής αρχιτεκτονικής που είδαμε παραπάνω και αποτελείται από τις εξής μονάδες (modules):



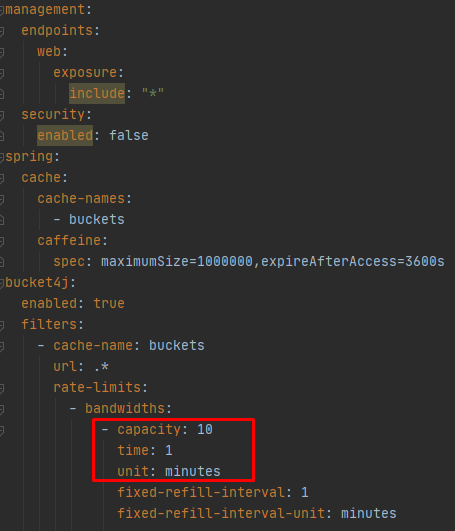
Σχήμα: Αρχιτεκτονική ενός crawler

1. Το Όριο URL (URL Frontier), που περιέχει διευθύνσεις URL που δεν έχουν ακόμη ληφθεί στην τρέχουσα ανίχνευση (στην περίπτωση συνεχούς ανίχνευσης, μια διεύθυνση URL μπορεί να έχει προηγουμένως ανακτηθεί αλλά επιστρέφει στο σύνορο για επανάκτηση).
2. Μία Μονάδα Επίλυσης DNS (DNS Resolution module) που καθορίζει τον διακομιστή ιστού από τον οποίο θα γίνει ανάκτηση της σελίδας που καθορίζεται από μια διεύθυνση URL.
3. Μία Μονάδα Ανάκτησης (Fetch Module) που χρησιμοποιεί το πρωτόκολλο http για να ανακτήσει την ιστοσελίδα σε α URL.
4. Μια Μονάδα Ανάλυσης (Parsing module) που εξάγει το κείμενο και το σύνολο των συνδέσμων από μία ανακτημένη ιστοσελίδα.
5. Μια Μονάδα Κατάργησης Διπλοτύπων (Duplicate Elimination module) που καθορίζει αν ένας σύνδεσμος που έχει εξαχθεί βρίσκεται ήδη στα σύνορα URL ή έχει ανακτηθεί πρόσφατα.

Captchas: Οι Captchas χρησιμοποιούνται στην απομάκρυνση του spam. Ωστόσο, αποτελούν επίσης μια μεγάλη πρόκληση προσβασιμότητας στα web crawling bots διότι διακόπτουν την εξαγωγή δεδομένων. Για αυτό, θα χρειαζόμαστε ένα ενδιάμεσο (ή ενσωματωμένο) λογισμικό που μπορεί να λάβει captcha, να το λύσει και να επιστρέψει την απάντηση.

Για την αντιμετώπιση του ζητήματος αυτού χρησιμοποιήθηκε βιβλιοθήκη bucket4j η οποία εφαρμόζει τον αλγόριθμο token bucket. Τα βασικά του αλγορίθμου είναι εύκολα κατανοητά. Έχετε έναν κάδο που περιέχει μέγιστο αριθμό διακριτικών (χωρητικότητα). Κάθε φορά που ένας καταναλωτής θέλει να καλέσει μια υπηρεσία ή να καταναλώσει έναν πόρο, παίρνει ένα ή περισσότερα κουπόνια. Ο καταναλωτής μπορεί να καταναλώσει μια υπηρεσία μόνο εάν μπορεί να πάρει τον απαιτούμενο αριθμό διακριτικών. Εάν ο κάδος δεν περιέχει τον απαιτούμενο αριθμό διακριτικών, πρέπει να περιμένει έως ότου υπάρχουν αρκετά κουπόνια στον κάδο. Όταν έχουμε κάποιον που βγάζει κουπόνια, χρειαζόμαστε επίσης κάποιον που βάζει κουπόνια στον κάδο. Ο ανεφοδιασμός δημιουργεί περιοδικά νέα κουπόνια και τα τοποθετεί στον κάδο.

Παρακάτω είναι οι ρυθμίσεις που θέσαμε στο προτζεκτ μας.



Όπως παρατηρούμε δίνουμε δικαίωμα στο χρήστη να εκτέλεσει έως 10 request\λεπτό.

# Injection prevention

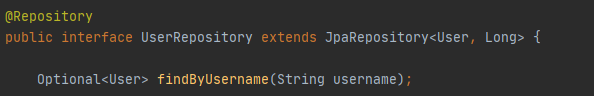
SQL Injection είναι μια από τις κορυφαίες 10 ευπάθειες εφαρμογών ιστού. Με απλά λόγια, το SQL Injection σημαίνει την injecting/inserting κώδικα SQL σε ένα ερώτημα μέσω δεδομένων που εισάγονται από τον χρήστη. Μπορεί να συμβεί σε οποιαδήποτε εφαρμογή χρησιμοποιώντας σχεσιακές βάσεις δεδομένων όπως Oracle, MySQL, PostgreSQL και SQL Server.

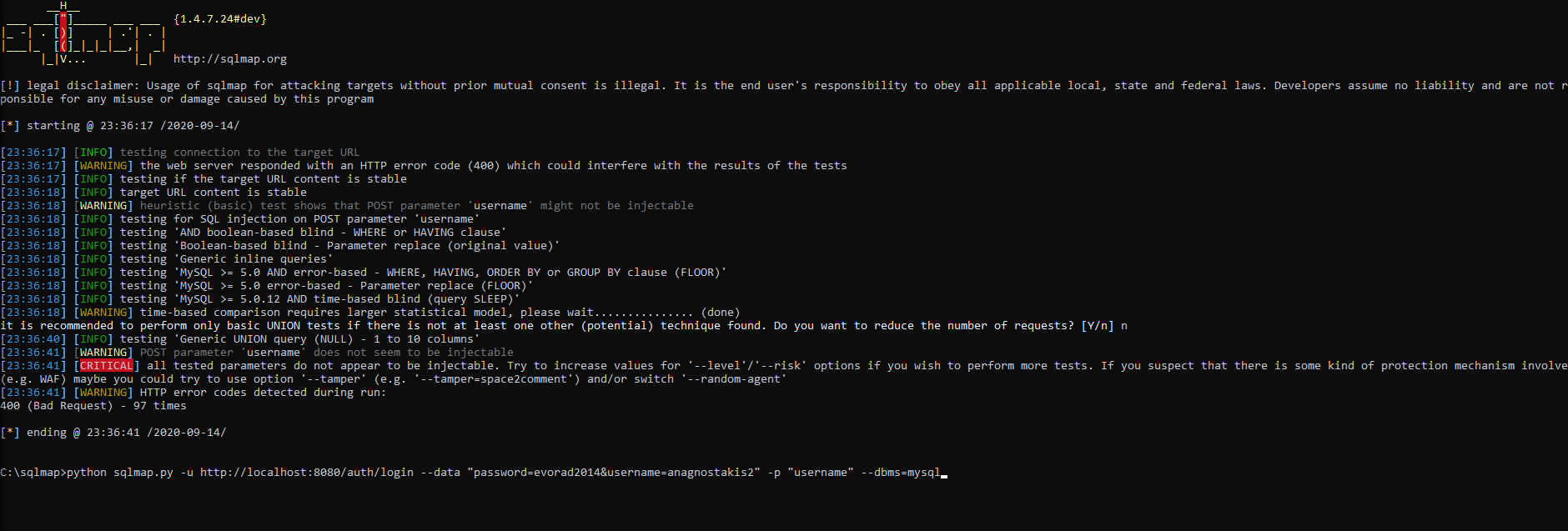
Για να εκτελέσει το SQL Injection, ένας κακόβουλος χρήστης προσπαθεί πρώτα να βρει μια θέση στην εφαρμογή όπου μπορεί να ενσωματώσει κώδικα SQL μαζί με δεδομένα. Μπορεί να είναι η σελίδα σύνδεσης οποιασδήποτε εφαρμογής ιστού ή οποιουδήποτε άλλου μέρους. Έτσι, όταν τα δεδομένα ενσωματωμένα με κώδικα SQL λαμβάνονται από την εφαρμογή, ο κώδικας SQL θα εκτελεστεί μαζί με το ερώτημα εφαρμογής.

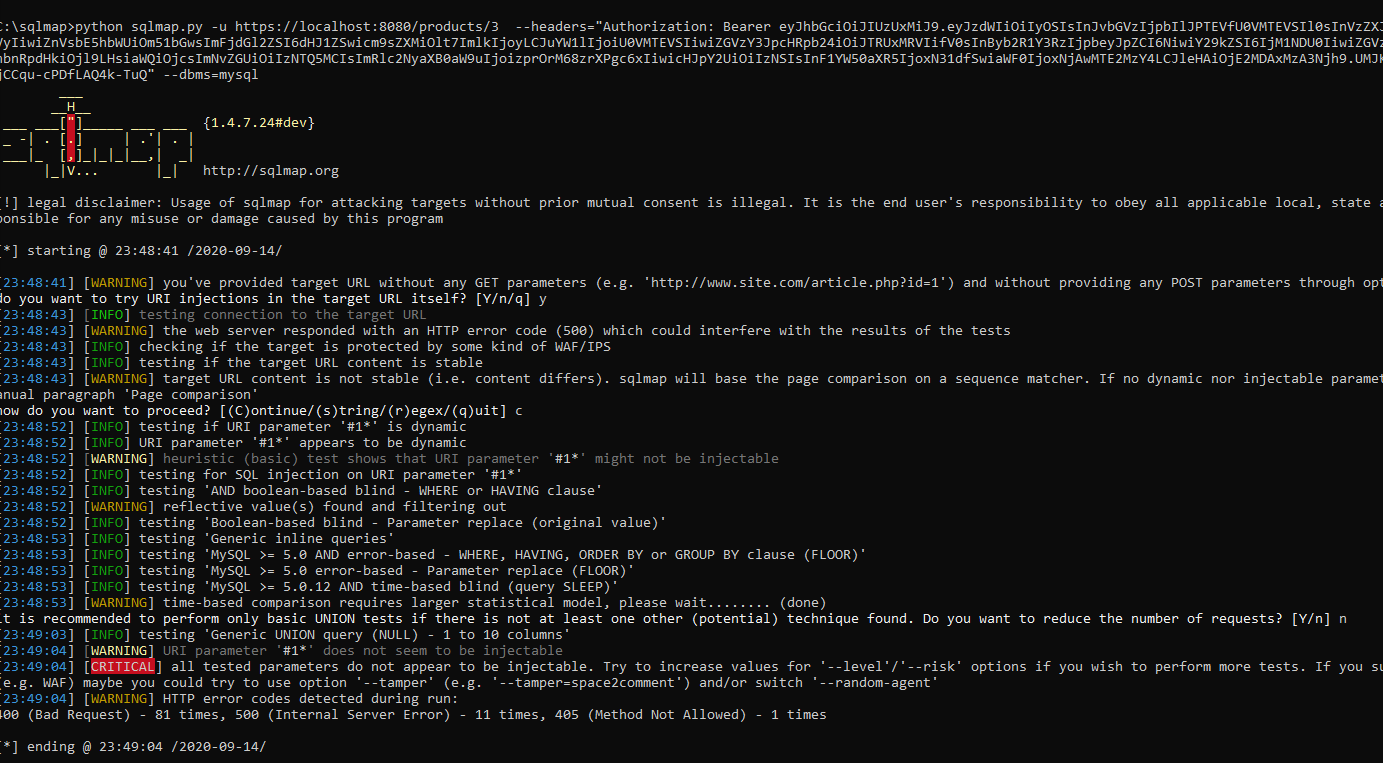
Αντίκτυπος του SQL Injection :

* Ένας κακόβουλος χρήστης μπορεί να αποκτήσει μη εξουσιοδοτημένη πρόσβαση στην εφαρμογή σας και να κλέψει δεδομένα.
* Μπορούν να αλλάξουν, να διαγράψουν δεδομένα στη βάση δεδομένων σας και να καταργήσουν την εφαρμογή σας.
* Ένας χάκερ μπορεί επίσης να πάρει τον έλεγχο του συστήματος στο οποίο εκτελείται ο διακομιστής βάσης δεδομένων εκτελώντας συγκεκριμένες εντολές συστήματος βάσης δεδομένων.

Για την αντιμετώπιση του SQL Injection χρησιμοποιήθηκε το JPA. Το Java Persistence API (JPA), είναι μια λύση ORM που αποτελεί μέρος του Java EE framework. Βοηθά στη διαχείριση σχεσιακών δεδομένων σε εφαρμογές που χρησιμοποιούν Java SE και Java EE. Είναι μια κοινή εσφαλμένη αντίληψη ότι οι λύσεις ORM όπως το JPA (Java Persistence API) είναι απόδειξη SQL Injection. Το JPA επιτρέπει τη χρήση της εγγενούς SQL και ορίζει τη δική του γλώσσα ερωτήματος, που ονομάζεται, JPQL (Java Persistence Query Language). Το πρώτο είναι επιρρεπές σε παραδοσιακές επιθέσεις έγχυσης SQL και το δεύτερο είναι επιρρεπές σε επιθέσεις έγχυσης JPQL (ή ORM).







# Session management

Τι είναι το Session management:

Το Session Management είναι ο έλεγχος των αλληλεπιδράσεων μεταξύ μιας εφαρμογής και χρηστών, ανάλογα με τη συμπεριφορά των πελατών σε πολλά αιτήματα.

Είναι γνωστό ότι το HTTP is a stateless protocol. Όλα τα αιτήματα και οι απαντήσεις είναι ανεξάρτητα. Ο διακομιστής δεν μπορεί να διακρίνει μεταξύ νέων επισκεπτών και επιστρεφόμενων επισκεπτών. Αλλά μερικές φορές ίσως χρειαστεί να παρακολουθούμε τη δραστηριότητα του πελάτη σε πολλά αιτήματα. Αυτό επιτυγχάνεται με τη διαχείριση συνεδρίας. Είναι ένας μηχανισμός που χρησιμοποιείται από το Web container για την αποθήκευση πληροφοριών περιόδου λειτουργίας για έναν συγκεκριμένο χρήστη.

Το Session management μπορεί να επιτευχθεί με τους εξής τρόπους:

* Cookies
* Hidden form field
* URL Rewriting
* HttpSession

Για να επιτύχουμε την σωστή διαχείριση των sessions χρησιμοποιήσαμε το JWT (JSON Web Token). Το JWT είναι η τυποποίηση για τη χρήση διακριτικών για έλεγχο ταυτότητας στον ιστό γενικά, όχι μόνο για υπηρεσίες REST. Είναι ανθεκτικό και μπορεί να μεταφέρει πολλές πληροφορίες, αλλά εξακολουθεί να είναι απλό στη χρήση, παρόλο που το μέγεθός του είναι σχετικά μικρό. Όπως και οποιοδήποτε άλλο διακριτικό, το JWT μπορεί να χρησιμοποιηθεί για να μεταβιβάσει την ταυτότητα των επικυρωμένων χρηστών μεταξύ ενός παρόχου ταυτότητας και ενός παρόχου υπηρεσιών (που δεν είναι απαραίτητα τα ίδια συστήματα). Μπορεί επίσης να φέρει όλη την αξίωση του χρήστη, όπως δεδομένα εξουσιοδότησης, οπότε ο πάροχος υπηρεσιών δεν χρειάζεται να μεταβεί στη βάση δεδομένων ή σε εξωτερικά συστήματα για να επαληθεύσει τους ρόλους και τα δικαιώματα χρήστη για κάθε αίτημα. Aυτά τα δεδομένα εξάγονται από το διακριτικό.

Εδώ είναι πώς σχεδιάζεται η ασφάλεια JWT:

Εικόνα που περιέχει στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

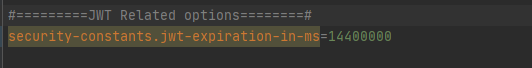
* Οι πελάτες συνδέονται αποστέλλοντας τα διαπιστευτήριά τους στον πάροχο ταυτότητας.
* Ο πάροχος ταυτότητας επαληθεύει τα διαπιστευτήρια. Eάν όλα είναι εντάξει, ανακτά τα δεδομένα χρήστη, δημιουργεί ένα JWT που περιέχει λεπτομέρειες και δικαιώματα του χρήστη που θα χρησιμοποιηθούν για την πρόσβαση στις υπηρεσίες και ορίζει επίσης τη λήξη στο JWT (το οποίο μπορεί να είναι απεριόριστο).
* Ο πάροχος ταυτότητας υπογράφει και, εάν χρειάζεται, κρυπτογραφεί το JWT και το στέλνει στον πελάτη ως απάντηση στο αρχικό αίτημα με διαπιστευτήρια.
* Ο πελάτης αποθηκεύει το JWT για περιορισμένο ή απεριόριστο χρονικό διάστημα, ανάλογα με τη λήξη που έχει ορίσει ο πάροχος ταυτότητας.
* Ο πελάτης στέλνει το αποθηκευμένο JWT σε μια κεφαλίδα εξουσιοδότησης για κάθε αίτημα στον πάροχο υπηρεσιών.
* Για κάθε αίτημα, ο πάροχος υπηρεσιών παίρνει το JWT από την Authorization κεφαλίδα και το αποκρυπτογραφεί, εάν χρειαστεί, επικυρώνει την υπογραφή και αν όλα είναι εντάξει, εξάγει τα δεδομένα χρήστη και τα δικαιώματα. Με βάση αυτά τα δεδομένα αποκλειστικά, και πάλι χωρίς να αναζητήσετε περισσότερες λεπτομέρειες στη βάση δεδομένων ή να επικοινωνήσετε με τον πάροχο ταυτότητας, μπορεί να αποδεχτεί ή να απορρίψει το αίτημα του πελάτη. Η μόνη απαίτηση είναι ότι η ταυτότητα και οι πάροχοι υπηρεσιών έχουν συμφωνία σχετικά με την κρυπτογράφηση, έτσι ώστε η υπηρεσία να μπορεί να επαληθεύσει την υπογραφή ή ακόμη και να αποκρυπτογραφήσει ποια ταυτότητα κρυπτογραφήθηκε.

Αυτή η ροή επιτρέπει μεγάλη ευελιξία, διατηρώντας παράλληλα τα πράγματα ασφαλή και εύκολο να αναπτυχθούν. Χρησιμοποιώντας αυτήν την προσέγγιση, είναι εύκολο να προσθέσετε νέους κόμβους διακομιστή στο σύμπλεγμα παρόχων υπηρεσιών, αρχικοποιώντας τους μόνο με τη δυνατότητα επαλήθευσης της υπογραφής και αποκρυπτογράφησης των διακριτικών παρέχοντάς τους ένα κοινόχρηστο μυστικό κλειδί. Δεν απαιτείται αναπαραγωγή συνεδρίας, συγχρονισμός βάσης δεδομένων ή επικοινωνία μεταξύ κόμβων.

Παρόλο που οι παλιές, τυποποιημένες προσεγγίσεις ασφαλείας (session cookies, HTTP Basic και HTTP Digest) θα λειτουργήσουν και με τις υπηρεσίες REST, όλοι έχουν προβλήματα που θα ήταν ωραίο να αποφευχθούν χρησιμοποιώντας ένα καλύτερο πρότυπο. Το JWT είναι ικανό να λύσει όλα τα προβλήματα.

Το κύριο πλεονέκτημα του JWT είναι ο χειρισμός του ελέγχου ταυτότητας χρηστών, διατηρώντας τα πάντα ασφαλή με τα ενημερωμένα πρότυπα κρυπτογράφησης. Η αποθήκευση αξιώσεων (ρόλοι χρήστη και δικαιώματα) στο ίδιο το διακριτικό δημιουργεί τεράστια οφέλη σε κατανεμημένες αρχιτεκτονικές συστήματος, όπου ο διακομιστής που εκδίδει το αίτημα δεν έχει πρόσβαση στην πηγή δεδομένων ελέγχου ταυτότητας.

Στο προτζεκτ μας χρησιμοποιήσαμε jwt token το οποίο περιέχει όλα τα απαραίτητα στοιχεία για την ομαλή λειτουργία του front μέρους. Οι ρυθμίσεις που δώσαμε στο jwt μας είναι οι παρακάτω.



Θέσαμε την λήξη του jwt στα 14400000 ms ή αλλιώς στις 4 ώρες.

# Server hardening

Ο όρος hardening είναι η διαδικασία παραμετροποίησης των ρυθμίσεων ασφαλείας ενός συστήματος με στόχο τη μείωση των ευπαθειών. Ένα σύστημα κατά κύριο λόγο είναι πιο ευπαθές ανάλογα με τον αριθμό των λειτουργιών που πραγματοποιεί. Όσο περισσότερες είναι οι λειτουργίες τόσο μεγαλύτερη είναι η πιθανότητα εμφάνισης κάποιας ευπάθειας. Με σκοπό την μείωση του κινδύνου, αποτελεί κοινή τακτική η αφαίρεση περιττού λογισμικού, περιττών χρηστών του συστήματος ή συνδέσεων και η απενεργοποίηση ή κατάργηση περιττών υπηρεσιών.

Υπάρχουν διαφορετικές προσεγγίσεις που μπορεί να ακολουθηθούν ώστε να κάνουμε harden ένα Linux λειτουργικό σύστημα. Αυτό μπορεί να περιλαμβάνει, μεταξύ άλλων, την ενσωμάτωση ενός patch ασφάλειας για τον πυρήνα, όπως το Exec Shield, το κλείσιμο των δικτυακών θυρών, την εγκατάσταση συστημάτων ανίχνευσης εισβολών και firewalls. Υπάρχουν επίσης, έτοιμα script και εργαλεία όπως το Linux Bastille που με φιλικό τρόπο προς τον χρήση παραμετροποιούν το σύστημα ανάλογα με της ανάγκες του.

Για την αντιμετώπιση των ευπαθειών του server χρησιμοποιήσαμε το πρωτόκολλο ασφαλείας SSL που προκύπτει από τους όρους “Secure Sockets Layer” και πρόκειται για ένα πρωτόκολλο επικοινωνίας με ηλεκτρονικό πιστοποιητικό, το οποίο εγγυάται την ασφαλή μεταφορά δεδομένων μεταξύ ενός server (website) και ενός browser (client). Πιο συγκεκριμένα, το πρωτόκολλο SSL είναι υπεύθυνο για:

* την πιστοποίηση του server από τον browser
* την παροχή ενός ασφαλούς κρυπτογραφημένου περιβάλλοντος για την ανταλλαγή δεδομένων μέσω αυτού.

Επομένως, η επικοινωνία με ένα website το οποίο δεν χρησιμοποιεί πρωτόκολλο SSL μπορεί οποιαδήποτε στιγμή να παραβιαστεί από κάποιον hacker, ο οποίος θα αποκτήσει αυτομάτως πρόσβαση σε όλα τα στοιχεία που μοιράζονται οι δύο πλευρές. Από emails, ονόματα και τηλεφωνικές επαφές μέχρι στοιχεία λογαριασμών και κωδικοί ασφαλείας, όλα μπορεί να γίνουν εύκολη λεία για κάποιον που δεν έχει αγνές προθέσεις.

Αντιθέτως, τα websites που είναι κρυπτογραφημένα με SSL προστατεύουν τους επισκέπτες τους, ώστε κανένας να μην μπορεί να έχει πρόσβαση στην επικοινωνία τους με τον εκάστοτε browser.

Ο διακομιστής HTTP Apache δεν απαιτεί εισαγωγή, καθώς είναι πιθανώς ο πιο διάσημος και δημοφιλής διακομιστής ιστού που υπάρχει. Είναι δυνατή η εκτέλεση του Apache πολύ εύκολα σε πολλές διαφορετικές πλατφόρμες και ρυθμίσεις. Η εφαρμογή συνοδεύεται από πολλές λειτουργικές μονάδες τρίτων μερών για τη διαχείριση διαφορετικών ειδών εργασιών (mod\_rewrite για επανεγγραφή διευθύνσεων URL βάσει κανόνα) και ένα από αυτά, είναι το mod\_proxy : Το Apache Module για την εφαρμογή ενός διακομιστή μεσολάβησης (ή πύλης) για διακομιστές που εκτελούνται στο back-end.

Το mod\_proxy είναι η μονάδα Apache για ανακατεύθυνση συνδέσεων (δηλαδή μια πύλη, που τις περνά). Είναι ενεργοποιημένο για χρήση όπως κάθε άλλη ενότητα και η διαμόρφωση είναι αρκετά βασική (ή τυπική), σύμφωνα με άλλες. Το mod\_proxy δεν είναι μόνο μία ενότητα αλλά μια συλλογή από αυτά, με το καθένα να φέρνει ένα νέο σύνολο λειτουργιών.

Μερικές από αυτές τις ενότητες είναι:

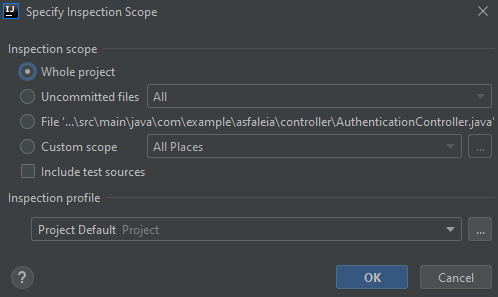
* **mod\_proxy:** Η κύρια μονάδα διακομιστή μεσολάβησης για το Apache που διαχειρίζεται τις συνδέσεις και τις ανακατευθύνει.
* **mod\_proxy\_http:** Αυτή η ενότητα εφαρμόζει τις δυνατότητες διακομιστή μεσολάβησης για πρωτόκολλα HTTP και HTTPS.
* **mod\_proxy\_ftp:** Αυτή η ενότητα κάνει το ίδιο, αλλά για πρωτόκολλο FTP.
* **mod\_proxy\_connect:** Αυτό χρησιμοποιείται για το πρωτόκολλο SSL.
* **mod\_proxy\_ajp:** Χρησιμοποιείται για εργασία με το πρωτόκολλο AJP.
* **mod\_proxy\_wstunnel:** Χρησιμοποιείται για εργασία με web-sockets (π.χ. WS και WSS).
* **mod\_proxy\_balancer:** Χρησιμοποιείται για ομαδοποίηση και εξισορρόπηση φορτίου.
* **mod\_cache:** Χρησιμοποιείται για προσωρινή αποθήκευση.
* **mod\_headers:** Χρησιμοποιείται για τη διαχείριση κεφαλίδων HTTP.
* **mod\_deflate:** Χρησιμοποιείται για συμπίεση.

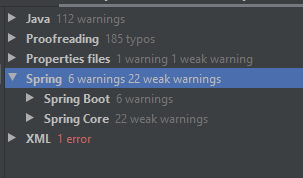
Στην εργασία μας χρησιμοποιήσαμε ssl για επικοινωνία από την angular στον apache καθώς και από τον apache στο Spring boot προτζεκτ μας.



# Code auditing

Λέγοντας code auditing εννοούμε την ανάλυση του κώδικα μας από τρίτο πρόγραμμα όπου εντοπίζονται σφάλματα, δυσλειτουργίες και κενά στον κωδικά μας. Για το προτζεκτ μας χρησιμποιήσαμε το ενσωματωμένο code auditing του intellij. Παρακάτω φαίνονται μερικά από τα απολέσματα μας.



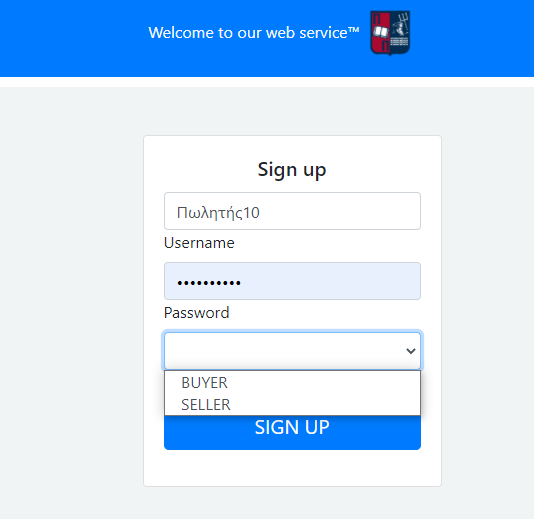


Από τα παραπάνω διορθώσαμε προβλήματα όπως deprecated μεθόδους που δεν χρησιμποιούνται πλέον στην java 8, null exceptions καθώς και μεταβλητές οι οποίες θα έπρεπε να είναι τύπου final ή private. Παράλληλα έγινε ένα είδος refactoring στον κώδικα όπου αφαιρέθηκαν unsude imports καθώς και σχόλια μέσα στον κώδικα.

# Παρουσίαση εφαρμογής

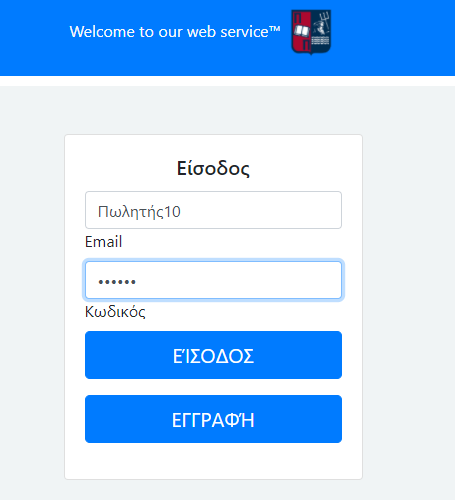
Registration

Ο χρήστης μπορεί να κάνει εγγραφη και να επιλέξει αν θα ειναι αγοραστής ή πωλητής.



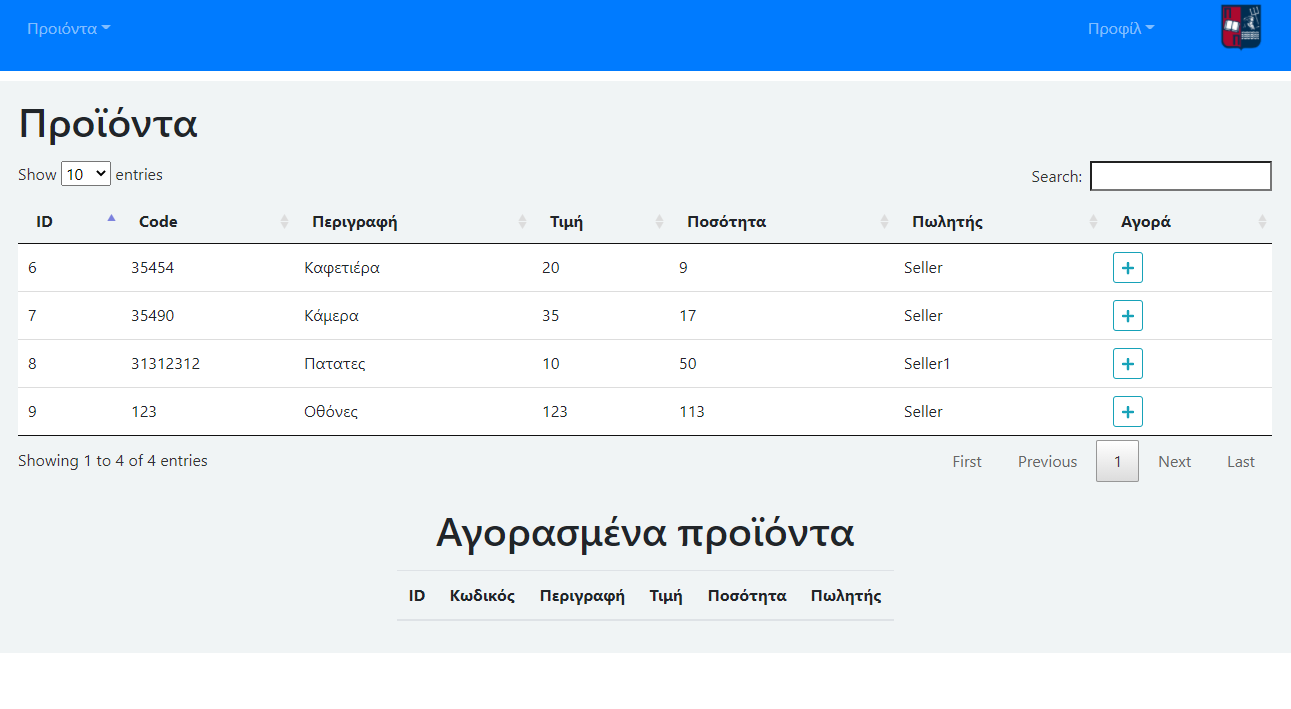
Login

Ο χρήστης εισάγει τα στοιχεία του και κανει login. Ανάλογα με το ρόλο του θα οδηγηθεί και στην αντίστοιχη σελίδα

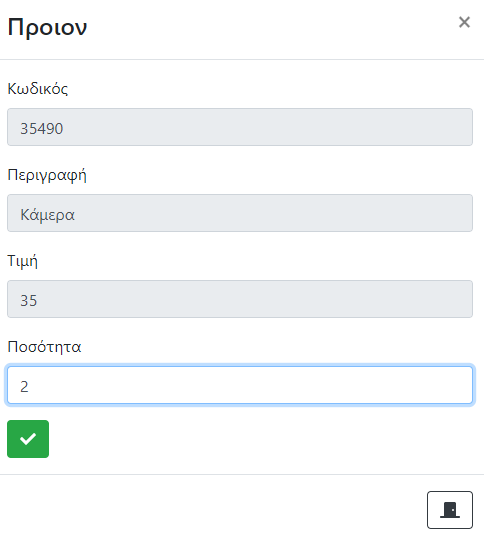


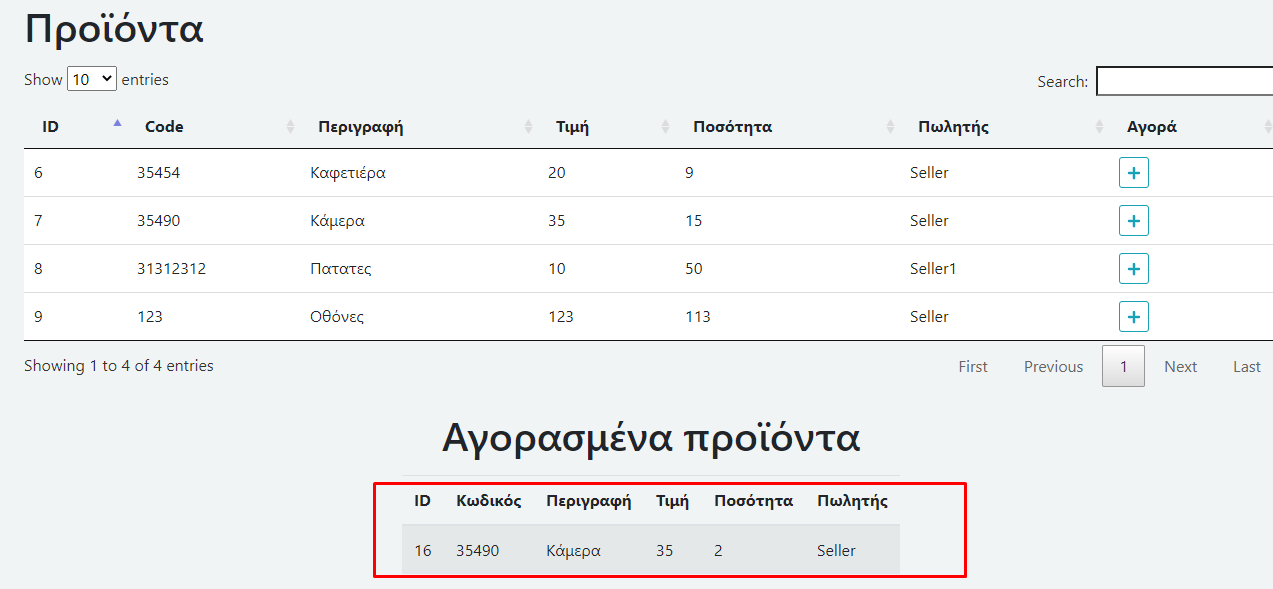
Αγορά προιόντων

Εδώ ο buyer βλέπει τα πράγματα πωλουνται και από κάτω τον πίνακα με αυτά που έχει αγοράσει.

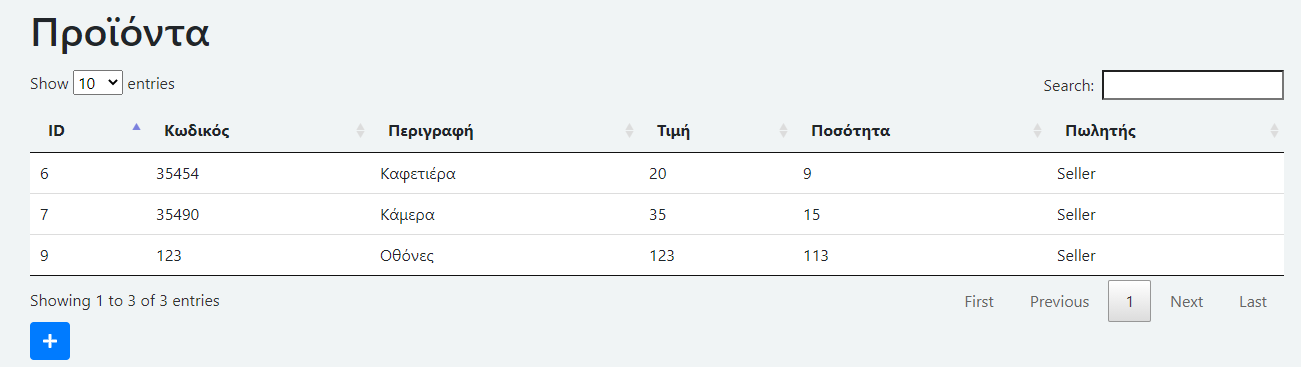


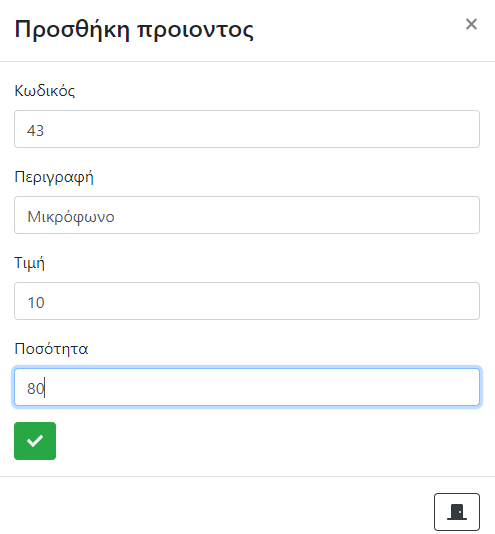
Στη συνέχεια μπορεί να επιλέξε «Αγορά» και να αγοράσει την ποσότητα που επιθυμεί.



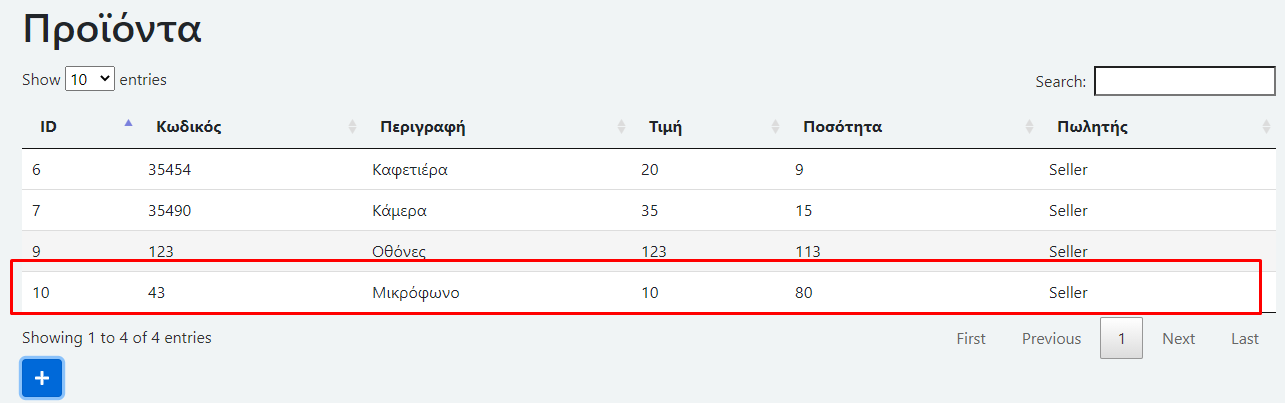


Αντίστοιχα ο αγοραστής οδηγείται σε αυτή τη σελίδα όπου βλέπει τα προιόντα που πουλάει και του δίνεται η δυνατότητα να προσθέσει επιπλέον προιοντα.





Στην εισαγωγή του προιόντος ανανεώνεται ο πίνακας



Ο admin οδηγείται στην σελίδα καλωσορίσματος.



Τέλος δίνεται το δικαίωμα ο χρήστης να κανει logout επιλέγοντας Προφίλ -> Έξοδος.

