



ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Ανάπτυξη REST API για mobile e-banking εφαρμογές
με την χρήση νέων τεχνολογιών.**

Μαυρουδής Στυλιανός

Εισηγητής: Δρ Κωνσταντίνος Κουκουλέτσος, Καθηγητής

ΑΘΗΝΑ
ΙΟΥΝΙΟΣ 2018

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**Ανάπτυξη REST API για mobile e-banking εφαρμογές
με την χρήση νέων τεχνολογιών.**

**Μαυρουδής Στυλιανός
Α.Μ. 0113**

Εισηγητής:

Δρ Κωνσταντίνος Κουκουλέτσος, Καθηγητής

Εξεταστική Επιτροπή:

.....
.....

Ημερομηνία εξέτασης

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος Μαυρουδής Στυλιανός, του Παναγιώτη, με αριθμό μητρώου ais-0113 φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο. Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης. Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφαση της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

(Κενό φύλλο)

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της ανάπτυξης εφαρμογής με νέες τεχνολογίες και την χρήση νέων τεχνολογιών υποδομής με σκοπό την βαριά χρήση της από εφαρμογές κινητού.

Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου , τον οποίο θα ήθελα να ευχαριστήσω.

Ακόμα θα ήθελα να ευχαριστήσω την οικογένειά μου, τους φίλους μου και όσους στάθηκαν και έκαναν υπομονή κοντά μου κατά την διάρκεια της συγγραφής της διπλωματικής.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

(Κενό φύλλο)

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη διαδικτυακής εφαρμογής και αρχιτεκτονικής API για βαρέως τύπου χρήση από εφαρμογές κινητού με σκοπό παροχής συνολικής πληροφορίας για όλους τους τραπεζικούς λογαριασμούς ενός χρήστη σε διαφορετικές τράπεζες. Παρουσιάζεται μια ολοκληρωμένη λύση για το παραπάνω πρόβλημα που φθάνει ακόμη και στην ανάλυση της αρχιτεκτονικής συστημάτων της υποδομής που θα υποστηρίξει την υλοποιημένη εφαρμογή.

ABSTRACT

This diploma thesis deals with the development of web application using REST API architecture style for heavy duty type use in mobile applications in order to provide aggregated information from different bank accounts for a bank customer. We present a complete solution to the above problem by analysing even the infrastructure architecture that will support the implemented application.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Ανάπτυξη API για mobile εφαρμογές
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: API, διαδίκτυο, Cloud, Docker, Symfony, AWS

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1	15
1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας	15
ΚΕΦΑΛΑΙΟ 2	17
2.1 Περιγραφή του Symfony Framework.....	17
2.2 Εγκατάσταση Symfony Framework.....	18
2.3 Δομή αρχείων Symfony Framework.....	22
2.4 Παραμετροποίηση Symfony Framework	24
2.5 Δημιουργία Bundles	25
2.6 Δημιουργία Controller	26
2.7 Δημιουργία Entities	29
2.8 Εντολές Symfony	35
2.9 Αυθεντικοποίηση χρήστη μέσω του API	45
ΚΕΦΑΛΑΙΟ 3	55
3.1 Περιγραφή της λειτουργίας του API της εφαρμογής	55
3.2 Περιγραφή της λειτουργίας της Δικτυακής εφαρμογής	60
ΚΕΦΑΛΑΙΟ 4	63
4.1 Περιγραφή του Cloud Computing	63
4.2 Οριζόντια και Κάθετη κλιμάκωση συστημάτων	64
4.3 Εικονικοποίηση (Virtualization).....	64
4.4 Εικονικοποίηση Πλατφόρμας.....	64
4.5 Περιγραφή του Docker	65
4.6 Εγκατάσταση Docker	66
4.7 Περιγραφή της σχεσιακής βάσης δεδομένων MySQL	67
4.8 Περιγραφή της γνώσης προγραμματισμού PHP.....	70
4.9 Περιγραφή του εξυπηρετητή ιστού Nginx	71
ΚΕΦΑΛΑΙΟ 5	73
5.1 Περιγραφή του διακομιστή υπηρεσιών cloud AWS	73
5.2 Amazon Relational Database Service (RDS).....	73
5.3 Elastic Compute Cloud (Amazon EC2).....	77
5.4 Amazon Simple Storage Service (Amazon S3).....	78
ΚΕΦΑΛΑΙΟ 6	79
6.1 Περιγραφή της μεθοδολογίας ανάπτυξης λογισμικού Test Driven Development	79
6.2 Περιγραφή της μεθοδολογίας Continuous Integration(CI).....	80
6.3 Δημιουργία s3 bucket.....	80
6.4 Δημιουργία χρήστη συστήματος	81
6.5 Δημιουργία εικονικών μηχανών.....	83
6.6 Δημιουργία κατανομητή φορτίου (Load Balancer).....	88
6.7 Παραμετροποίηση της υπηρεσίας CodeDeploy του AWS	91
6.8 Σύνδεση Git με την υπηρεσία CodeDeploy της Amazon	94
ΣΥΜΠΕΡΑΣΜΑΤΑ	97
ΒΙΒΛΙΟΓΡΑΦΙΑ	99

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Εικόνα 1 Περιγραφή χρήσης του API.....	15
Εικόνα 2 Αρχική σελίδα μετά την εγκατάσταση του Symfony	20
Εικόνα 3 Σελίδα ελέγχου εγκατάστασης Symfony	21
Εικόνα 4 Δομή αρχείων Symfony	23
Εικόνα 5 Χρήση εφαρμογής Postmanγια την αποστολή αιτήσεων στην εφαρμογή	51
Εικόνα 6 Αυθεντικοποίηση χρήστη μέσω API	52
Εικόνα 7 Αίτηση στοιχείων χρήστη από το API	53
Εικόνα 8 Εισαγωγή χρήστη μέσω της διαδουκτιακής εφαρμογής.....	60
Εικόνα 9 Λίστα τραπεζικών λογαριασμών μέσω δικτυακής εφαρμογής.....	60
Εικόνα 10 Λίστα καρτών χρήστη μέσω διαδικτυακής εφαρμογής.....	61
Εικόνα 11 Κινήσεις λογαριασμών χρήστη μέσω διαδικτυακής εφαρμογής	61
Εικόνα 12 Λίστα τραπεζών	61
Εικόνα 13 Σχήμα της βάσης δεδομένων της εφαρμογής	68
Εικόνα 14 Περιγραφή συνδιασμού χρήσης MySql Nginx PHP/FPM.....	72
Εικόνα 15 Δημιουργία RDS - βήμα 1	73
Εικόνα 16 Δημιουργία RDS βήμα 2	74
Εικόνα 17 Δημιουργία RDS βήμα 3	75
Εικόνα 18 Δημιουργία RDS βήμα 4	75
Εικόνα 19 Δημιουργία RDS βήμα 5	76
Εικόνα 20 Δημιουργία RDS βήμα 6	77
Εικόνα 21 Έλεγχος απομακρυσμένης σύνδεσης με το RDS.....	77
Εικόνα 22 Περιγραφή μεθοδολογίας TDD.....	79
Εικόνα 23 Δημιουργία S3 βήμα 1	81
Εικόνα 24 Λίστα s3.....	81
Εικόνα 25 Δημιουργία χρήστη συστήματος βήμα 1.....	82
Εικόνα 26 Δημιουργία χρήστη συστήματος βήμα 2.....	82
Εικόνα 27 Έλεγχος στοιχείων του χρήστη συστήματος.....	83
Εικόνα 28 Δημιουργία Auto Scaling Group βήμα 1	83
Εικόνα 29 Επιλογή τύπου εικονικής μηχανής	84

Εικόνα 30	Επιλογή χαρακτηριστικών εικονικής μηχανής.....	84
Εικόνα 31	Δημιουργία Launch Configuration Βήμα 1	85
Εικόνα 32	Δημιουργία Launch Configuration βήμα 2	85
Εικόνα 33	Δημιουργία Launch Configuration βήμα 2	86
Εικόνα 34	Χρήση κρυπτογραφημένου αρχείου pem	86
Εικόνα 35	Δημιουργία Auto Scaling Group Βήμα 1	87
Εικόνα 36	Δημιουργία Auto Scaling Group βήμα 2	87
Εικόνα 37	Δημιουργία καταναμητή φορτίου βήμα 1	88
Εικόνα 38	Δημιουργία καταναμητή φορτίου βήμα 2	88
Εικόνα 39	Δημιουργία καταναμητή φορτίου βήμα 3	89
Εικόνα 40	Δημιουργία καταναμητή φορτίου βήμα 4	89
Εικόνα 41	Δημιουργία καταναμητή φορτίου βήμα 5	90
Εικόνα 42	Παραμετροποίηση της υπηρεσίας CodeDeploy βήμα 1.....	91
Εικόνα 43	Παραμετροποίηση υπηρεσίας CodeDeploy βήμα 2.....	91
Εικόνα 44	Παραμετροποίηση υπηρεσίας CodeDeploy βήμα 3.....	92
Εικόνα 45	Παραμετροποίηση υπηρεσίας CodeDeploy βήμα 4.....	93
Εικόνα 46	Μεταβλητές περιβάλλοντος για την διασύνδεση Bitbucket με CodeDeploy.....	94
Εικόνα 47	Deployment σε εξέλιξη από την πλατφόρμα του Bitbucket.....	95
Εικόνα 48	Deployment σε εξέλιξη από την AWS κονσόλα.....	96

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 1 Λίστα Controller της εφαρμογής	30
Πίνακας 2 Αντιστοίχιση μοντέλων με πίνακες στην βάση δεδομένων	34

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

REST	Representational state transfer
API	Application Programming Interface
AWS	Amazon Web Services
S3	Simple Storage Solution
EC2	Elastic Compute Cloud
ORM	Object-relational mapping

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

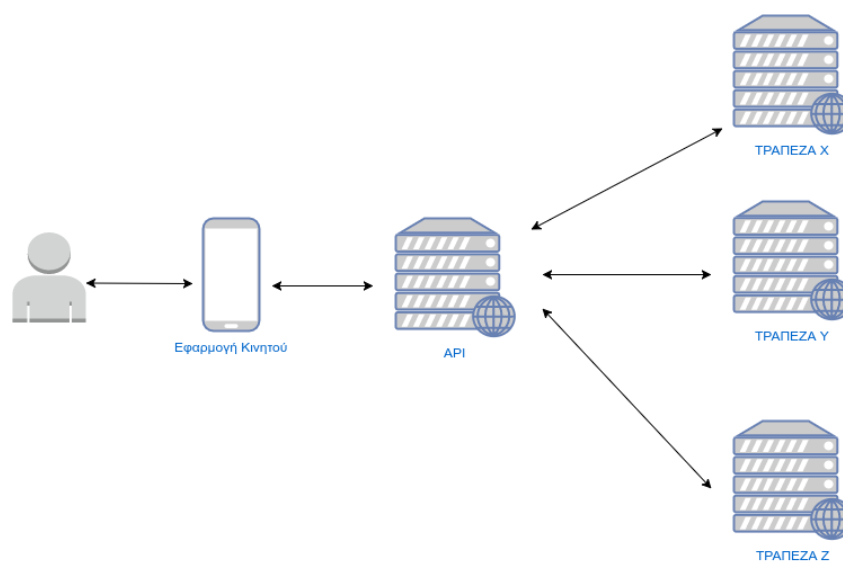
ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Η παρούσα πτυχιακή εργασία έχει σκοπό να παρουσιάσει την υλοποίηση ενός λογισμικού REST API (Representational state transfer Application programming interface) για την χρήση του κυρίως από εφαρμογές κινητών για την ταυτόχρονη επικοινωνία με πολλά διαφορετικά τραπεζικά συστήματα με την χρήση νέων τεχνολογιών.

Συγκεκριμένα θα μπορεί μια mobile εφαρμογή αφού είναι εξουσιοδοτημένη από το API να πραγματοποιεί αιτήσεις (Requests) για την άντληση αθροιστικής πληροφορίας από τους διάφορους τραπεζικούς λογαριασμούς ενός χρήστη, όπως για παράδειγμα το συνολικό διαθέσιμο υπόλοιπο χρημάτων που έχει κάποιος σε διαφορετικούς τραπεζικούς λογαριασμούς



Εικόνα 1 Περιγραφή χρήσης του API

Με την χρήση των νέων τεχνολογιών, cloud υπηρεσιών, σύγχρονων frameworks και την εφαρμογή μοντέρνων αρχιτεκτονικών η ανάπτυξη μιας εφαρμογής μπορεί να γίνει με γρηγορότερο και ασφαλέστερο τρόπο με στόχο το σύστημα να είναι πάντα σταθερό και να μπορεί να εξυπηρετήσει τεράστιο όγκο από χρήστες. Ένα REST API δεν χρησιμοποιείται μόνο από mobile εφαρμογές αλλά και από οποιοδήποτε είδος εφαρμογών και για την επικοινωνία κυρίως μεταξύ συστημάτων.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

ΚΕΦΑΛΑΙΟ 2

Symfony Framework

2.1 Περιγραφή του Symfony Framework

Για την υλοποίηση του API χρησιμοποιήθηκε το Symfony 3 framework για την γλώσσα προγραμματισμού PHP. Το Symfony είναι ένα από τα καλύτερα και πιο ολοκληρωμένα frameworks της γλώσσας PHP με την πρώτη του εμφάνιση του στα τέλη του 2005 υπό την άδεια του MIT και με αρχικό δημιουργό τον Fabien Potencier και την Sensio Labs.

Ένα framework περιέχει ένα σύνολο από βιβλιοθήκες με ενιαίο τρόπο γραφής με στόχο την επαναχρησιμοποίησή τους. Φτιάχνοντας μια ομάδα από μηχανικούς λογισμικού που γνωρίζουν από κοινού το ίδιο framework υπάρχει καλύτερη επικοινωνία μεταξύ των μελών της με αποτέλεσμα τα καλύτερα και ποιοτικότερα αποτελέσματα για την υλοποίηση ενός έργου καθώς και γίνεται πιο εύκολη η προσθήκη νέων μελών στην ομάδα.

Η χρήση ενός framework για την υλοποίηση ενός λογισμικού δεν είναι αναγκαστική αποτελεί όμως βασική προϋπόθεση για την χρήση ή μη καθώς και την επιλογή του κατάλληλου framework, κατά την ανάλυση και σχεδίαση του λογισμικού πριν την έναρξη της ανάπτυξής του.

Η σωστή σχεδίαση της αρχιτεκτονικής ενός λογισμικού φέρνει ως αποτέλεσμα την ταχύτερη και ευκολότερη υλοποίησή του, έτσι επιλέγοντας ένα framework είναι σαν να έχουμε ήδη διαλέξει μια βάση που περιέχει έτοιμες, βέλτιστες και σταθερές λύσεις για κοινά θέματα που χρειάζεται να έχει ένα λογισμικό όπως για παράδειγμα την επικοινωνία της εφαρμογής με τις βάσεις δεδομένων.

Για το τελευταίο παράδειγμα το Symfony διαθέτει το Doctrine ORM (Object-relational mapping) για την επικοινωνία με την σχεσιακή βάση δεδομένων αλλά και για την αναπαράσταση των δεδομένων σε οντότητες.

Παρακάτω θα περιγράψουμε το συγκεκριμένο ORM και θα δούμε παραδείγματα σύνδεσης με την βάση δεδομένων MySQL από το υλοποιημένο API της πτυχιακής.

Ένα ακόμα κοινό χαρακτηριστικό και από τα βασικότερα ζητήματα που έχουμε να λύσουμε κατά την ανάπτυξη ενός λογισμικού είναι η πιστοποίηση (authentication) ενός χρήστη στο σύστημα μας όσο το δυνατότερο ασφαλέστερη γίνεται για την μέγιστη προστασία των δεδομένων ενός χρήστη αλλά και του ίδιου του συστήματος.

Μετά την πιστοποίηση ενός χρήστη σε ένα σύστημα θα πρέπει να έχουμε προβλέψει για την κατάλληλη εξουσιοδότηση (authorization) επιλογών που θα έχει ένας πιστοποιημένος χρήστης. Για παράδειγμα ένας διαχειριστής συστήματος έχει διαφορετικές δυνατότητες από έναν απλό πιστοποιημένο χρήστη.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

2.2 Εγκατάσταση Symfony Framework

Μπορούμε να εγκαταστήσουμε το Symfony με τους εξής παρακάτω τρόπους:

1. Να κατεβάσουμε το zip αρχείο με περιεχόμενα τα απαραίτητα αρχεία για την ανάπτυξη μιας web εφαρμογής με το Symfony framework.
2. Να κάνουμε clone το repository του Symfony από το Github.
3. Να χρησιμοποιήσουμε το εργαλείο Symfony installer.
4. Να χρησιμοποιήσουμε το εργαλείο composer.

Παρακάτω θα δούμε τον τέταρτο τρόπο εγκατάστασης του Symfony χρησιμοποιώντας δηλαδή το εργαλείο composer. Για την εγκατάσταση του Symfony θα πρέπει να είμαστε σίγουροι ότι έχουμε επίσης εγκατεστημένα στο σύστημά μας μια έκδοση της γλώσσας PHP και το εργαλείο composer.

Το εργαλείο composer αποτελεί τον διαχειριστή εξάρτησης πακέτων της PHP (Dependency Manager for PHP). Στην συνέχεια θα αναλύσουμε με ποιόν τρόπο θα έχουμε διαθέσιμα τόσο την PHP όσο και το εργαλείο του composer.

Αρχικοποίηση του έργου μας με composer:

```
composer create-project symfony/website-skeleton my-project
```

Με την παραπάνω εντολή θα δημιουργηθεί ένας φάκελος `my-project` και στην συνέχεια θα εγκατασταθούν τα κατάλληλα πακέτα και βιβλιοθήκες καθώς θα χτιστεί ο σκελετός του έργου μας για να μπορεί να είναι λειτουργικό. Μετά το τέλος της αρχικοποίησης θα εμφανιστούν στην οθόνη μας οδηγίες για τα επόμενα βήματα.

```
ms86@ms86-jupiter ~$ composer create-project symfony/website-skeleton my-project
Cannot create cache directory /home/ms86/.composer/cache/repo/https---packagist.org/ or directory is not writable. Proceeding without cache
Installing symfony/website-skeleton (v3.4.6)
Cannot create cache directory /home/ms86/.composer/cache/files/ or directory is not writable. Proceeding without cache
- Installing symfony/website-skeleton (v3.4.6)
  Downloading: 100%
Created project in my-project
Cannot create cache directory /home/ms86/.composer/cache/repo/https---packagist.org/ or directory is not writable. Proceeding without cache
Cannot create cache directory /home/ms86/.composer/cache/files/ or directory is not writable. Proceeding without cache
Loading composer repositories with package information
Updating dependencies (including require-dev)
- Installing ocramius/package-versions (1.2.0)
  Downloading: 100%
- Installing symfony/flex (v1.0.78)
  Downloading: 100%
Cannot create cache directory /home/ms86/.composer/cache/repo/https---symfony.sh/ or directory is not writable. Proceeding without cache
Some Symfony Flex features may not work as expected: your version of Composer is too old
Please upgrade using https://getcomposer.org/
Cannot create cache directory /home/ms86/.composer/cache/repo/https---packagist.org/ or directory is not writable. Proceeding without cache
- Installing doctrine/lexer (v1.0.1)
  Downloading: 100%
- Installing doctrine/inflexor (v1.2.0)
  Downloading: 100%
- Installing doctrine/collections (v1.4.0)
  Downloading: 100%
- Installing doctrine/cache (v1.6.2)
  Downloading: 100%
- Installing doctrine/annotations (v1.4.0)
  Downloading: 100%
- Installing doctrine/common (v2.7.3)
  Downloading: 100%
- Installing symfony/polyfill-mbstring (v1.7.0)
  Downloading: 100%
```

```
Writing lock file
Generating autoload files
ocramius/package-versions: Generating version class...
ocramius/package-versions: ...done generating version class
Symfony operations: 21 recipes (7180d7ced5d85910795e67b53df7b20a)
 - Configuring symfony/flex (>=1.0): From github.com/symfony/recipes:master
 - Configuring symfony/framework-bundle (>=3.3): From github.com/symfony/recipes:master
 - Configuring doctrine/annotations (>=1.0): From github.com/symfony/recipes:master
 - Configuring symfony/webpack-encore-pack (>=1.0): From github.com/symfony/recipes:master
 - Configuring symfony/translation (>=3.3): From github.com/symfony/recipes:master
 - Configuring symfony/twig-bundle (>=3.3): From github.com/symfony/recipes:master
 - Configuring symfony/swiftmailer-bundle (>=2.5): From github.com/symfony/recipes:master
 - Configuring symfony/security-bundle (>=3.3): From github.com/symfony/recipes:master
 - Configuring symfony/routing (>=3.3): From github.com/symfony/recipes:master
 - Configuring symfony/console (>=3.3): From github.com/symfony/recipes:master
 - Configuring doctrine/doctrine-cache-bundle (>=1.3.3): From auto-generated recipe
 - Configuring doctrine/doctrine-bundle (>=1.6): From github.com/symfony/recipes:master
 - Configuring doctrine/doctrine-migrations-bundle (>=1.2): From github.com/symfony/recipes:master
 - Configuring symfony/monolog-bundle (>=3.1): From github.com/symfony/recipes:master
 - Configuring sensio/framework-extra-bundle (>=4.0): From github.com/symfony/recipes:master
 - Configuring symfony/web-profiler-bundle (>=3.3): From github.com/symfony/recipes:master
 - Configuring easycorp/easy-log-handler (>=1.0): From github.com/symfony/recipes:master
 - Configuring symfony/debug-bundle (>=3.3): From github.com/symfony/recipes:master
 - Configuring symfony/maker-bundle (>=1.0): From github.com/symfony/recipes:master
 - Configuring symfony/phpunit-bridge (>=3.3): From github.com/symfony/recipes:master
 - Configuring symfony/web-server-bundle (>=3.3): From github.com/symfony/recipes:master
Executing script cache:clear [OK]
Executing script assets:install --symlink --relative public [OK]

Some files may have been created or updated to configure your new packages.
Please review, edit and commit them: these files are yours.

What's next?

* Run your application:
  1. Change to the project directory
  2. Execute the php -S 127.0.0.1:8000 -t public command;
  3. Browse to the http://localhost:8000/ URL.

  Quit the server with CTRL-C.
  Run composer require server --dev for a better web server.

* Read the documentation at https://symfony.com/doc

Database Configuration

* Modify your DATABASE_URL config in .env

* Configure the driver (mysql) and
  server_version (5.7) in config/packages/doctrine.yaml

How to test?

* Write test cases in the tests/ folder
* Run php bin/phpunit
```

Μπορούμε να δοκιμάσουμε ότι όλα έχουν πάει καλά με την αρχικοποίηση της web εφαρμογή μας τρέχοντας έναν server για PHP για χρήση μόνο κατά την διάρκεια της ανάπτυξης της εφαρμογής. Σε επόμενο κεφάλαιο θα δούμε ποιόν web server θα χρησιμοποιήσουμε αλλά και με ποιόν τρόπο.

Με την παρακάτω εντολή εγκαθιστούμε και τρέχουμε ένα απλό server που παρέχει το Symfony:

- `cd my-project`
- `composer require server --dev`
- `php bin/console server:run`

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

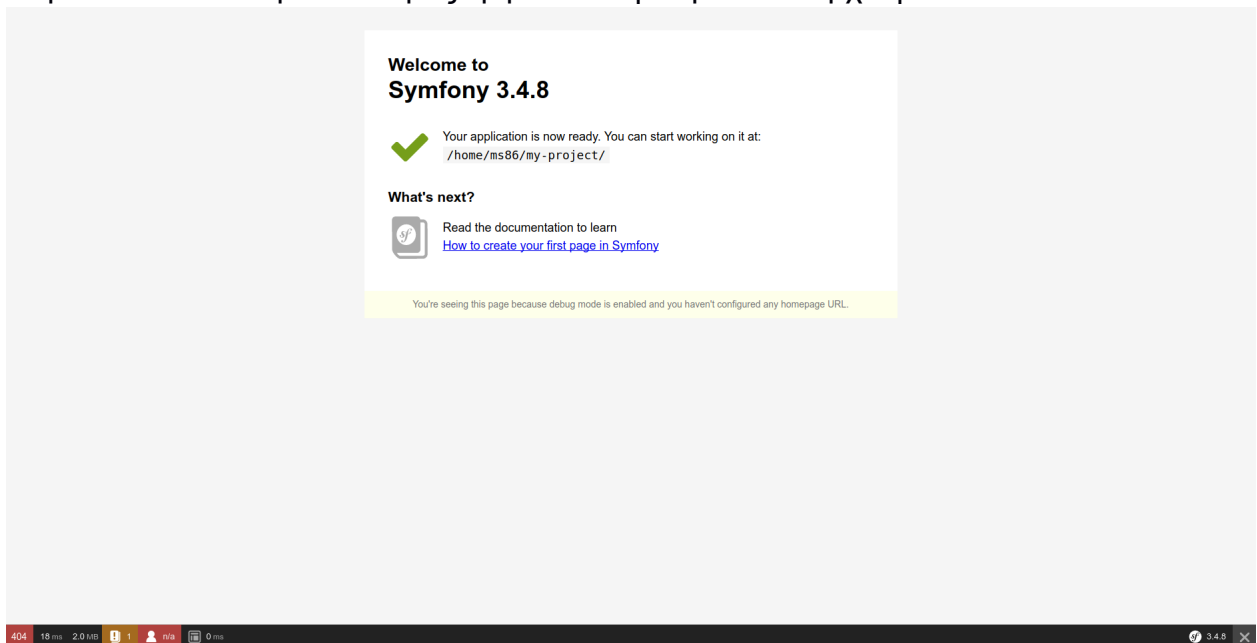
```
ms86@ms86-jupiter ~ - /my-project - master php bin/console server:run

[OK] Server Listening on http://127.0.0.1:8000

// Quit the server with CONTROL-C.

PHP 7.0.28-0ubuntu0.16.04.1 Development Server started at Sun Apr 15 16:27:00 2018
Listening on http://127.0.0.1:8000
Document root is /home/ms86/my-project/public
Press Ctrl-C to quit.
[Sun Apr 15 16:27:11 2018] 127.0.0.1:52760 [404]: /
[Sun Apr 15 16:27:12 2018] 127.0.0.1:52764 [200]: /_wdt/f7906a
```

Ο web server τρέχει με επιτυχία και ακούει στην πόρτα 8000, έτσι ακολουθώντας το παρακάτω url θα πρέπει να μας εμφανιστεί η παρακάτω αρχική εικόνα:



Εικόνα 2 Αρχική σελίδα μετά την εγκατάσταση του Symfony

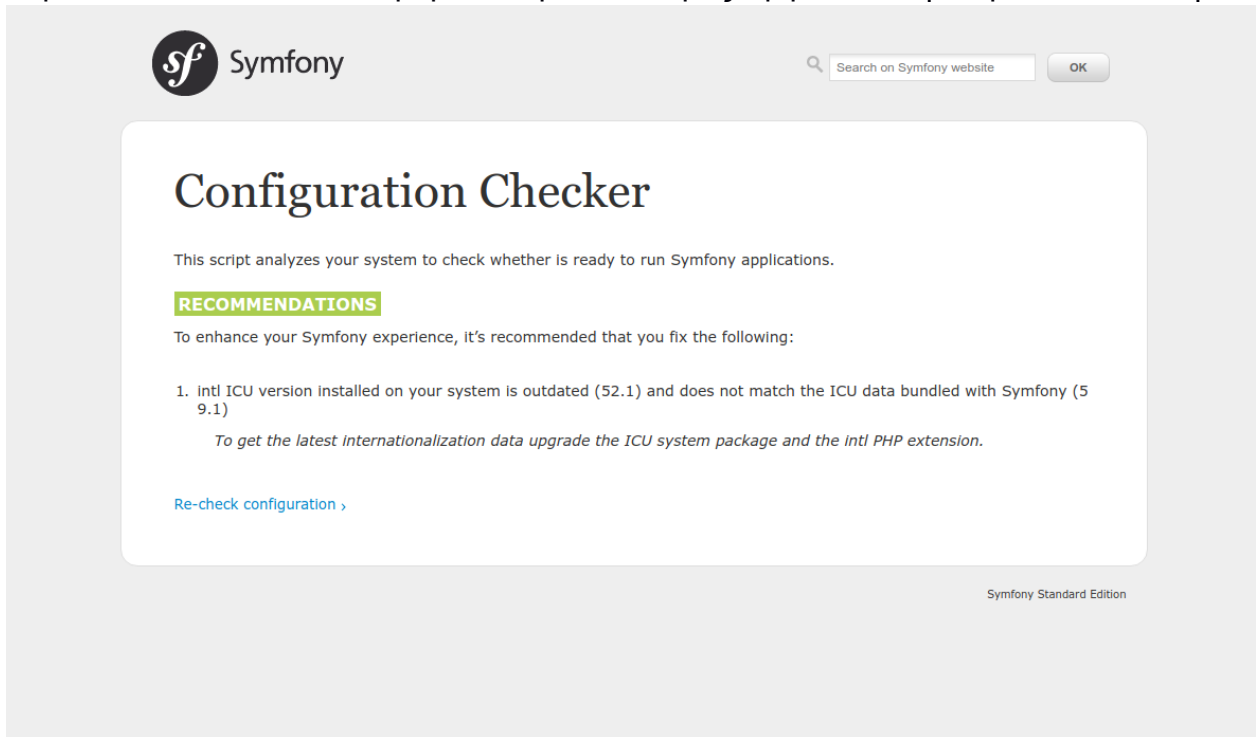
Ένα ακόμα χρήσιμο πακέτο για το Symfony php framework είναι το `symfony/requirements-checker` με το οποίο μπορούμε να ελέγξουμε εάν το σύστημά μας πληρεί όλες τις προϋποθέσεις για να υποστηρίξει ένα Symfony web application.

Με την παρακάτω εντολή εγκαθιστούμε το `symfony/requirements-checker`:

- `cd your-project/`
- `composer require symfony/requirements-checker`

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Μετά την εγκατάσταση του πακέτου και ακολουθώντας τον σύνδεσμο <http://localhost:8000/check.php> θα πρέπει να μας εμφανιστεί η παρακάτω οθόνη:



Εικόνα 3 Σελίδα ελέγχου εγκατάστασης Symfony

Αν υπάρχουν σημαντικά λάθη ή ελλείψεις από το σύστημα τότε αυτά επισημαίνονται με κόκκινο χρώμα στην συγκεκριμένη σελίδα.

2.3 Δομή αρχείων Symfony Framework

Για την υλοποίηση του InfoMyMoney API της διπλωματικής εργασίας χρησιμοποιήθηκε η έκδοση 3.3.10 του Symfony και παρακάτω φαίνεται και περιγράφεται η δομή και ο σκελετός των αρχείων της εφαρμογής : [2]

```
ms86@ms86-jupiter ~/tei_pir_ptuxiaki_api/app > ls -la
total 176
drwxrwxr-x 9 ms86 ms86 4096 Ιαν 6 01:52 .
drwxrwxr-x 7 ms86 ms86 4096 Απρ 12 16:51 ..
drwxrwxr-x 4 ms86 ms86 4096 Ιαν 6 01:46 app
drwxrwxr-x 2 ms86 ms86 4096 Ιαν 6 01:56 bin
-rw-rw-r-- 1 ms86 ms86 2688 Ιαν 6 01:39 composer.json
-rw-rw-r-- 1 ms86 ms86 121739 Ιαν 6 01:39 composer.lock
-rw-rw-r-- 1 ms86 ms86 265 Οκτ 6 2017 .gitignore
-rw-rw-r-- 1 ms86 ms86 1065 Οκτ 6 2017 LICENSE
-rw-rw-r-- 1 ms86 ms86 988 Οκτ 6 2017 phpunit.xml.dist
-rw-rw-r-- 1 ms86 ms86 2423 Οκτ 6 2017 README.md
drwxrwxr-x 4 ms86 ms86 4096 Οκτ 26 21:22 src
drwxrwxr-x 4 ms86 ms86 4096 Οκτ 26 21:22 tests
drwxrwxr-x 6 ms86 ms86 4096 Ιαν 6 01:56 var
drwxrwxr-x 25 ms86 ms86 4096 Ιαν 6 01:56 vendor
drwxrwxr-x 3 ms86 ms86 4096 Απρ 15 16:09 web
ms86@ms86-jupiter ~/tei_pir_ptuxiaki_api/app > ls -l app
total 16
-rw-rw-r-- 1 ms86 ms86 101 Οκτ 6 2017 AppCache.php
-rw-rw-r-- 1 ms86 ms86 2166 Ιαν 6 01:46 AppKernel.php
drwxrwxr-x 2 ms86 ms86 4096 Ιαν 6 02:15 config
drwxrwxr-x 4 ms86 ms86 4096 Οκτ 25 01:46 Resources
ms86@ms86-jupiter ~/tei_pir_ptuxiaki_api/app > ls -l bin
total 8
-rwxr-xr-x 1 ms86 ms86 837 Οκτ 6 2017 console
-rwxr-xr-x 1 root root 3935 Ιαν 6 01:56 symfony_requirements
ms86@ms86-jupiter ~/tei_pir_ptuxiaki_api/app > ls -l src
total 8
drwxr-xr-x 6 ms86 ms86 4096 Ιαν 5 15:26 ApiBundle
drwxrwxr-x 8 ms86 ms86 4096 Οκτ 27 00:58 AppBundle
ms86@ms86-jupiter ~/tei_pir_ptuxiaki_api/app > ls -l web
total 56
-rw-rw-r-- 1 ms86 ms86 1170 Οκτ 22 17:22 app_dev.php
-rw-rw-r-- 1 ms86 ms86 2092 Οκτ 6 2017 apple-touch-icon.png
-rw-rw-r-- 1 ms86 ms86 645 Οκτ 6 2017 app.php
drwxrwxr-x 2 ms86 ms86 4096 Ιαν 6 01:56 bundles
-rw-rw-rw- 1 ms86 ms86 21488 Απρ 15 16:09 config.php
-rw-rw-r-- 1 ms86 ms86 6518 Οκτ 6 2017 favicon.ico
-rw-rw-r-- 1 ms86 ms86 104 Οκτ 22 03:23 index.php
-rw-rw-r-- 1 ms86 ms86 116 Οκτ 6 2017 robots.txt
ms86@ms86-jupiter ~/tei_pir_ptuxiaki_api/app > ls -ls app/config
total 40
4 -rw-rw-r-- 1 ms86 ms86 1170 Ιαν 5 17:15 config_dev.yml
4 -rw-rw-r-- 1 ms86 ms86 511 Οκτ 6 2017 config_prod.yml
4 -rw-rw-r-- 1 ms86 ms86 270 Οκτ 6 2017 config_test.yml
4 -rw-rw-r-- 1 ms86 ms86 3648 Ιαν 6 02:15 config.yml
4 -rw-rw-r-- 1 ms86 ms86 344 Ιαν 6 01:56 parameters.yml
4 -rw-rw-r-- 1 ms86 ms86 1139 Ιαν 6 01:30 parameters.yml.dist
4 -rw-rw-r-- 1 ms86 ms86 328 Οκτ 6 2017 routing_dev.yml
4 -rw-rw-r-- 1 ms86 ms86 363 Οκτ 27 02:02 routing.yml
4 -rw-rw-r-- 1 ms86 ms86 2234 Ιαν 6 02:09 security.yml
4 -rw-rw-r-- 1 ms86 ms86 1883 Ιαν 5 17:46 services.yml
ms86@ms86-jupiter ~/tei_pir_ptuxiaki_api/app >
```

- **app/config** : Περιέχει όλες τις κοινές ρυθμίσεις αλλά και τις διαφορετικές ρυθμίσεις για κάθε περιβάλλον.

- **app/Resources** : Περιέχει όλα τα αρχεία που αφορούν την εμφάνιση του application καθώς και τα αρχεία που είναι υπεύθυνα για την μετάφραση του περιεχομένου
- **Bin**: Περιέχει όλα τα αρχεία που μπορούν να εκτελεστούν από το την κονσόλα του συστήματος ως εντολές.
- **Src**: Περιέχει όλα τα αρχεία που αφορούν την λογική της εφαρμογής μας.
- **Web**: Είναι ο φάκελος που περιέχει το αρχείο index.php όπου περνάνε όλες οι αιτήσεις της εφαρμογής, καθώς και όλα τα βοηθητικά αρχεία που αφορούν την λειτουργικότητα και την εμφάνιση της εφαρμογής προς το χρήστη.
- **Vendor**: Περιέχει όλες τις βιβλιοθήκες και τα πακέτα στα οποία εξαρτάται το Symfony για να είναι λειτουργικό και η εγκατάσταση των οποίων γίνεται μέσω του εργαλείου composer.
- **var/cache**: Περιέχει όλα τα προσωρινά αρχεία που δημιουργούνται από την εφαρμογή στον δίσκο για γρηγορότερη προσπέλαση.
- **var/logs**: Περιέχει όλη την καταγραφή συμβάντων/σφαλμάτων που πραγματοποιούνται στην εφαρμογή.
- **var/sessions**: Περιέχει τα αρχεία σχετικά με την συνεδρία του χρήστη στην εφαρμογή.
- **Tests**: Περιέχει όλα τα αυτοματοποιημένα τεστ για την σωστή λειτουργία της εφαρμογής.

```
1  blog/  
2  └─ app/  
3     └─ config/  
4         Resources/  
5  └─ bin  
6     └─ console  
7  └─ src/  
8     └─ AppBundle/  
9  └─ var/  
10     └─ cache/  
11         logs/  
12         sessions/  
13  └─ tests/  
14     └─ AppBundle/  
15  └─ vendor/  
16  └─ web/
```

Εικόνα 4 Δομή αρχείων Symfony

2.4 Παραμετροποίηση Symfony Framework

Το Symfony χρησιμοποιεί αρχεία yml για να μπορέσουμε να το παραμετροποιήσουμε. [2]

Στον φάκελο config βρίσκονται όλα τα αρχεία παραμετροποίησης του Symfony όπως φαίνονται παρακάτω:

```
→ tei_pir_ptuxiaki_api git:(master) X ls -lr app/app/config
total 40
-rw-r--r-- 1 stelios stelios 1883 Apr 29 15:12 services.yml
-rw-r--r-- 1 stelios stelios 2234 Apr 29 15:12 security.yml
-rw-r--r-- 1 stelios stelios 363 Apr 29 15:12 routing.yml
-rw-r--r-- 1 stelios stelios 328 Apr 29 15:12 routing_dev.yml
-rw-r--r-- 1 stelios stelios 1139 Apr 29 15:12 parameters.yml.dist
-rw-rw-r-- 1 stelios stelios 344 Maï 1 13:06 parameters.yml
-rw-r--r-- 1 stelios stelios 3652 Maï 1 13:08 config.yml
-rw-r--r-- 1 stelios stelios 270 Apr 29 15:12 config_test.yml
-rw-r--r-- 1 stelios stelios 511 Apr 29 15:12 config_prod.yml
-rw-r--r-- 1 stelios stelios 1170 Apr 29 15:12 config_dev.yml
```

Το βασικότερο αρχείο είναι το **config.yml** στο οποίο γίνονται οι συνολικές ρυθμίσεις για το κάθε κομμάτι της εφαρμογής αντίστοιχα, όπως για παράδειγμα την σύνδεση με την βάση δεδομένων.

Αν για παράδειγμα θα χρειαστούμε να κάνουμε σύνδεση με τη βάση δεδομένων **MySQL** θα πρέπει προσθέσουμε τις ρυθμίσεις που αφορούν τον **driver** που θα χρησιμοποιήσει το application για αυτή την σύνδεση, τη διεύθυνση της βάσης δεδομένων καθώς και τα συνθηματικά και την πόρτα που ακούει ο mysql server όπως φαίνεται παρακάτω:

```
# Doctrine Configuration
doctrine:
  dbal:
    driver: pdo_mysql
    host: '%database_host%'
    port: '%database_port%'
    dbname: '%database_name%'
    user: '%database_user%'
    password: '%database_password%'
    charset: UTF8
```

Οι παραπάνω τιμές για την παραμετροποίηση της σύνδεσης της εφαρμογής με την βάση δεδομένων βρίσκονται ανάμεσα σε '%' όπου σημαίνει ότι παίρνουν τις τιμές τους από την αντίστοιχη μεταβλητή. Δηλαδή η τιμή του '%database_password%' παίρνει την τιμή της από την μεταβλητή database_password που βρίσκεται στο αρχείο parameters.yml.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Αυτό συμβαίνει για να μπορούμε να κάνουμε ίδια παραμετροποίηση σε διαφορετικά περιβάλλοντα αν για παράδειγμα η εφαρμογή μας εκτελείται σε δοκιμαστικό περιβάλλον που θα είναι προσβάσιμο μόνο από τα μέλη της ανάπτυξης της εφαρμογής θα πρέπει να έχει ίδιες ρυθμίσεις με αυτές που θα έχει η εφαρμογή στο περιβάλλον που θα είναι προσβάσιμο προς τους χρήστες/πελάτες της εφαρμογής (production περιβάλλον), αλλά να χρησιμοποιεί διαφορετικούς πόρους.

Έτσι για παράδειγμα η βάση δεδομένων που θα χρησιμοποιεί η εφαρμογή θα είναι διαφορετική ανά περιβάλλον και τυχόν λάθη δεν επηρεάσουν τα πραγματικά δεδομένα των χρηστών κατά την ανάπτυξη του έργου.

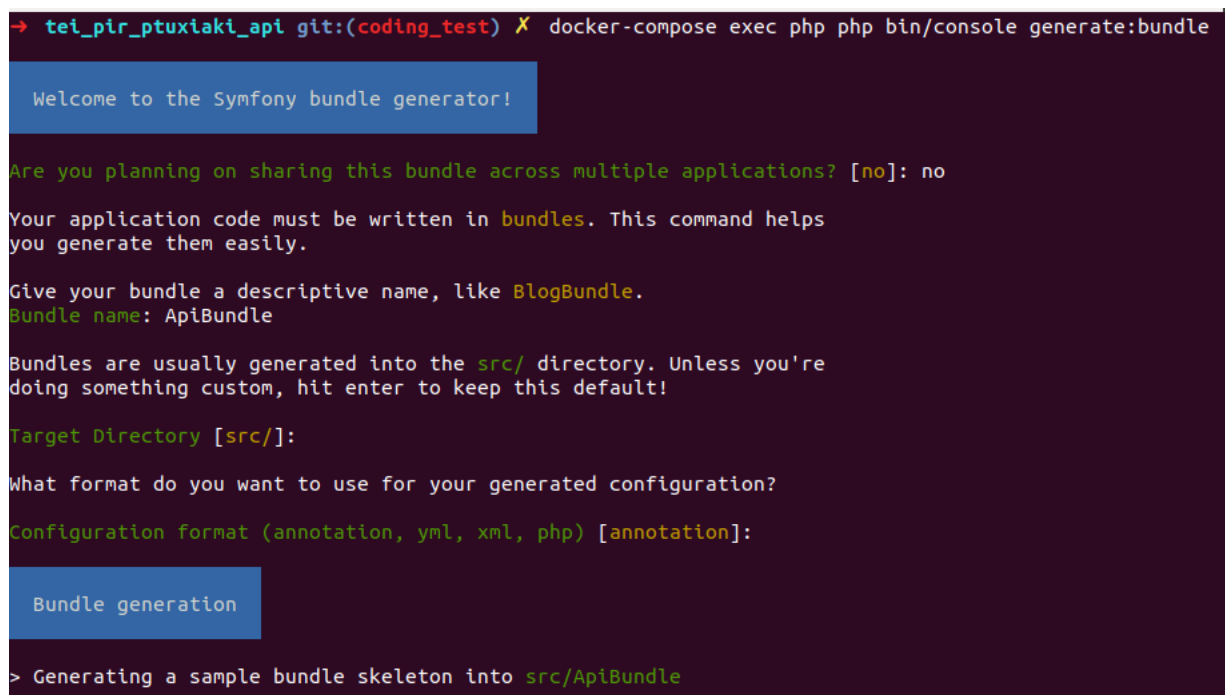
2.5 Δημιουργία Bundles

Ένας τρόπος να οργανώσουμε τον κώδικα της εφαρμογής μας με το Symfony είναι τα Bundles. [17]

Αποτελούν μιας μορφή επαναχρησιμοποιήσιμης βιβλιοθήκης μόνο που περιέχει όλα αυτά που χρειάζονται για να λειτουργήσει αυτόνομα μέσα στο Symfony, όπως οι Controllers μαζί με τους κανόνες για τις διαδρομές (Routes), τα δικά του μοντέλα/κλάσεις (Entities/Classes) υλοποιώντας την δική του λογική το καθένα και άλλα χαρακτηριστικά που μπορούν να χρησιμοποιηθούν από το Symfony.

Για να δημιουργήσουμε ένα Symfony Bundle τρέχουμε την παρακάτω εντολή όπου βήμα βήμα θα πρέπει να ορίσουμε το όνομα του Bundle μας, την τοποθεσία που θα δημιουργηθούν τα αρχεία αλλά και τον τρόπο με τον οποίο θα γίνεται η παραμετροποίηση αυτού:

```
docker-compose exec php php bin/console generate:bundle
```



```
→ tei_pir_ptuxiaki_api git:(coding_test) X docker-compose exec php php bin/console generate:bundle

Welcome to the Symfony bundle generator!

Are you planning on sharing this bundle across multiple applications? [no]: no

Your application code must be written in bundles. This command helps
you generate them easily.

Give your bundle a descriptive name, like BlogBundle.
Bundle name: ApiBundle

Bundles are usually generated into the src/ directory. Unless you're
doing something custom, hit enter to keep this default!

Target Directory [src/]:

What format do you want to use for your generated configuration?
Configuration format (annotation, yml, xml, php) [annotation]:

Bundle generation

> Generating a sample bundle skeleton into src/ApiBundle
```

Στην συγκεκριμένη εφαρμογή χρειάστηκε να δημιουργήσουμε δυο Bundles, το **AppBundle** το οποίο περιέχει όλα τα μοντέλα (**Entities**) μαζί με την λογική αλλά και τις εντολές της εφαρμογής και το **ApiBundle** που περιέχει συγκεκριμένα τους **Controllers** και την λογική που αφορούν το μέρος μόνο του **API** της εφαρμογής. Τα δυο αυτά Bundles βρίσκονται στο φάκελο src.

2.6 Δημιουργία Controller

Οι Controllers είναι οι οντότητες/κλάσεις που καθορίζουν, δέχονται και επεξεργάζονται τις αιτήσεις (Requests) και εμφανίζουν ανάλογα το ζητούμενο αποτέλεσμα, είτε αυτό είναι η εμφάνιση πληροφοριών σε μια ιστοσελίδα (webpage) είτε αποτελέσματα με συγκεκριμένη δομή υπό μορφή json στην περίπτωση μας.

Για να δημιουργήσουμε έναν Controller τρέχουμε την παρακάτω εντολή: [15]

```
docker-compose exec php php bin/console generate:controller
```

Ο χωρισμός και η ονοματοδοσία των Controllers γίνεται ανάλογα με τον σκοπό που εξυπηρετούν συνήθως όπως και επίσης θα πρέπει να μην περιέχουν κομμάτια λογικής που αφορούν το επιχειρηματικό μοντέλο (Business logic).

Για τα δυο Bundles της εφαρμογής μας που αναφέραμε στη προηγούμενη ενότητα (AppBundle και ApiBundle) έχουν δημιουργηθεί οι αντίστοιχοι Controllers για να διαχειριστούν τις αιτήσεις που αφορούν την κάθε οντότητα στην εφαρμογή όπως για παράδειγμα τους χρήστες.

UsersController	Για τις αιτήσεις που αφορούν τους χρήστες
AccountsController	Για τις αιτήσεις που αφορούν τους λογαριασμούς χρηστών.
CardsController	Για τις αιτήσεις που αφορούν τις κάρτες των χρηστών.
TransactionsController	Για τις αιτήσεις που αφορούν τις κινήσεις λογαριασμών των χρηστών.

Πίνακας 1 Λίστα Controller της εφαρμογής

Παρακάτω φαίνεται η δομή ενός Controller και συγκεκριμένα του **UsersController** όπου με την χρήση συγκεκριμένης μορφής σχολίων (Annotations) μπορούμε να ορίσουμε την διαδρομή του συνδέσμου καθώς και το αν ο σύνδεσμος είναι διαθέσιμος για χρήστες που δεν έχουν κάνει είσοδο στην εφαρμογή και άλλα.

Για παράδειγμα για τον σύνδεσμο του API όπου οι χρήστες μπορούν να κάνουν Login τα αντίστοιχα σχόλια στην μέθοδο **postLoginAction** είναι `* @Rest\Post("/Login")` που σημαίνει ότι :

1. το όνομα του συνδέσμου θα είναι της μορφής `http://localhost:8080/app_dev.php/api/users/login`
2. δέχεται μόνο POST Requests
3. και είναι προσβάσιμο από χρήστες που δεν έχουν πραγματοποιήσει είσοδο στο σύστημα.

Αντίστοιχα η μέθοδος **getTotalBalance** με annotation :

```
@Rest\Get("/totalBalance")
@Security("is_authenticated()")
```

που γυρνάει ως αποτέλεσμα το συνολικό διαθέσιμο υπόλοιπο από όλους τους λογαριασμούς ενός χρήστη σημαίνει ότι :

1. το όνομα του συνδέσμου θα είναι της μορφής `http://localhost:8080/app_dev.php/api/users/totalBalance`
2. δέχεται μόνο GET Requests
3. και είναι προσβάσιμο μόνο από χρήστες που έχουν κάνει είσοδο στην εφαρμογή.

ApiBundle/UsersApiController

```
/**
 * @Route("/users")
 * @Security("is_anonymous() or is_authenticated()")
 */
class UsersController extends Controller
{
    /**
     * @var UserPasswordEncoderInterface
     */
    private $passwordEncoder;

    /**
     * @var JwtEncoderInterface
     */
    private $jwtEncoder;

    public function __construct(UserPasswordEncoderInterface $passwordEncoder,
        JwtEncoderInterface $jwtEncoder)
    {
        $this->passwordEncoder = $passwordEncoder;
        $this->jwtEncoder = $jwtEncoder;
    }

    /**
     * @Rest\Post("/Login")
     */
    public function postLoginAction(Request $request)
    {
        $username = $request->getUser();
        $password = $request->getPassword();
    }
}
```

```
$user = $this->getDoctrine()
    ->getRepository('AppBundle:User')
    ->findOneBy(['username' => $username]);

if (!$user) {
    throw new BadCredentialsException();
}

$isPasswordValid = $this->passwordEncoder->isPasswordValid($user, $password);

if (!$isPasswordValid) {
    throw new BadCredentialsException();
}

$token = $this->jwtEncoder->encode(
    [
        'username' => $user->getUsername(),
        'exp'      => time() + 3600,
    ]
);

return new JsonResponse(['token' => $token]);
}

/**
 * @Rest\Get("/user")
 * @Security("is_authenticated()")
 */
public function getAction()
{
    $restresult = $this->getDoctrine()->getRepository('AppBundle:User')->find($this-
>getUser()->getId());

    if ($restresult === null) {
        return new View("An error occurred", Response::HTTP_NOT_FOUND);
    }

    return $restresult;
}

/**
 * @Rest\Get("/info")
 * @Security("is_authenticated()")
 */
public function getInfoAction()
{
    $restresult = $this->getDoctrine()
        ->getRepository('AppBundle:User')
        ->find($this->getUser()->getId());

    if ($restresult === null) {
        return new View("An error occurred", Response::HTTP_NOT_FOUND);
    }
    return $restresult;
}
```

```
/**
 * @Rest\Get("/totalBalance")
 * @Security("is_authenticated()")
 */
public function getTotalBalanceAction()
{
    $restresult = $this->getDoctrine()
        ->getRepository('AppBundle:Account')
        ->getTotalBalanceForUser($this->getUser()->getId());

    if ($restresult === null) {
        return new View("An error occurred", Response::HTTP_NOT_FOUND);
    }
    return reset($restresult);
}
}
```

2.7 Δημιουργία Entities

Τα Μοντέλα/Κλάσεις που αντιπροσωπεύουν τις οντότητες της πραγματικής ζωής και οι ιδιότητες αυτών αντιστοιχούν στον αντίστοιχο πίνακα μιας βάσης δεδομένων στο Symfony ονομάζονται Entities. [19]

Κάθε Entity έχει τα δικά του χαρακτηριστικά και την δική του λογική καθώς και εξαρτήσεις/συσχέτιση (Relation) από άλλα Entities. Η κλάση Account για παράδειγμα αποτελεί μια οντότητα (Entity) που αντιπροσωπεύει τον τραπεζικό λογαριασμό ενός χρήστη. Όπως ένας τραπεζικός λογαριασμός έχει κάποιον ιδιοκτήτη έτσι και η οντότητα Account έχει συσχέτιση με κάποιον χρήστη του συστήματος.

Για να δημιουργήσουμε μια καινούρια οντότητα τρέχουμε την παρακάτω εντολή:

```
docker-compose exec php php bin/console generate:entity
```

Οι οντότητες που δημιουργήθηκαν για την εφαρμογή είναι:

Οντότητα	Περιγραφή	Πίνακας βάσης δεδομένων
User	Αντιπροσωπεύει τους χρήστες της εφαρμογής.	fos_user
UserInfo	Αντιπροσωπεύει παραπάνω από τα βασικά στοιχεία ενός χρήστη.	user_info
Bank	Αντιπροσωπεύει τις διαθέσιμες τράπεζες.	bank
Transaction	Αντιπροσωπεύει τις συναλλαγές ενός λογαριασμού.	transaction

Account	Αντιπροσωπεύει τους τραπεζικούς λογαριασμούς	account
Card	Αντιπροσωπεύει τις κάρτες ενός χρήστη.	card

Πίνακας 2 Αντιστοίχιση μοντέλων με πίνακες στην βάση δεδομένων

Παρακάτω φαίνεται η δομή μιας οντότητας (Entity) και συγκεκριμένα της κλάσης **Account**.

```
/**
 * Account
 *
 * @ORM\Table(name="account")
 * @ORM\Entity(repositoryClass="AppBundle\Repository\AccountRepository")
 */
class Account
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @var int
     *
     * @ORM\Column(name="user_id", type="integer")
     */
    protected $userId;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string")
     */
    protected $name;

    /**
     * @var int
     *
     * @ORM\Column(name="type", type="integer")
     */
    protected $type;

    /**
     * @var int
     */
}
```

```
*
* @ORM\Column(name="bank_id", type="integer")
*
*/
protected $bank_id;

/**
* @var int
*
* @ORM\Column(name="number", type="integer")
*
*/
protected $number;

/**
* @var string
*
* @ORM\Column(name="iban", type="string")
*
*/
protected $iban;

/**
* @var double
*
* @ORM\Column(name="available_balance", type="float")
*
*/
protected $availableBalance;

/**
* @var float
*
* @ORM\Column(name="accounting_balance", type="float")
*
*/
protected $accountingBalance;

/**
* @var string
*
* @ORM\Column(name="friendly_name", type="string")
*
*/
protected $friendlyName;

/**
* @var AppBundle\Entity\User
*
* @ORM\ManyToOne(
*     targetEntity="\AppBundle\Entity\User"
* )
* @ORM\JoinColumn(name="user_id", referencedColumnName="id")
*
*/
protected $owner;
```



```
/**
 * Get id
 *
 * @return int
 */
public function getId()
{
    return $this->id;
}
/**
 * Set id
 *
 * @return int
 */
public function setId($id)
{
    $this->id = $id;
}

/**
 * @return int
 */
public function getType()
{
    return $this->type;
}

/**
 * @param int $type
 */
public function setType($type)
{
    $this->type = $type;
}

/**
 * @return int
 */
public function getBankId()
{
    return $this->bank_id;
}

/**
 * @param int $bank_id
 */
public function setBankId($bank_id)
{
    $this->bank_id = $bank_id;
}

/**
 * @return int
 */
public function getNumber()
{
```

```
        return $this->number;
    }
    /**
     * @param int $number
     */
    public function setNumber($number)
    {
        $this->number = $number;
    }

    /**
     * @return string
     */
    public function getIban()
    {
        return $this->iban;
    }

    /**
     * @param string $iban
     */
    public function setIban($iban)
    {
        $this->iban = $iban;
    }

    /**
     * @return float
     */
    public function getAvailableBalance()
    {
        return $this->availableBalance;
    }

    /**
     * @param float $availableBalance
     */
    public function setAvailableBalance($availableBalance)
    {
        $this->availableBalance = $availableBalance;
    }

    /**
     * @return float
     */
    public function getAccountingBalance()
    {
        return $this->accountingBalance;
    }

    /**
     * @param float $accountingBalance
     */
    public function setAccountingBalance($accountingBalance)
    {
        $this->accountingBalance = $accountingBalance;
    }
}
```

```
}

/**
 * @return User
 */
public function getOwner()
{
    return $this->owner;
}

/**
 * @param User $owner
 */
public function setOwner($owner)
{
    $this->owner = $owner;
}

/**
 * @return string
 */
public function getName()
{
    return $this->name;
}

/**
 * @param string $name
 */
public function setName($name)
{
    $this->name = $name;
}

/**
 * @return int
 */
public function getUserId()
{
    return $this->userId;
}

/**
 * @param int $userId
 */
public function setUserId($userId)
{
    $this->userId = $userId;
}

/**
 * @return mixed
 */
public function getFriendlyName()
{
    return $this->friendlyName;
}
```

```
/**
 * @param mixed $friendlyName
 */
public function setFriendlyName($friendlyName)
{
    $this->friendlyName = $friendlyName;
}
}
```

2.8 Εντολές Symfony

Το Symfony χρησιμοποιεί το Console component για να μπορεί να τρέξει σενάρια κώδικα/εντολές από το τερματικό (console comands). [1]

Το CLI (Command-line interface) του Symfony διαθέτει ήδη προ εγκατεστημένες αρκετές υλοποιημένες και πολύ βασικές εντολές αλλά μας βοηθάει να δημιουργήσουμε και τις δικές μας εντολές. Για να έχουμε πρόσβαση στις εντολές θα πρέπει να τρέξουμε στο τερματικό μας την παρακάτω εντολή όπου και θα μας εμφανιστεί η λίστα με όλες τις διαθέσιμες εντολές του Symfony μαζί και με τις δικές μας αν έχουμε δημιουργήσει.

```
php bin/console
```

```
→ tei_pir_ptuxiaki_api git:(master) X docker-compose run --rm --user www-data php bin/console
Symfony 3.3.10 (kernel: app, env: dev, debug: true)

Usage:
  command [options] [arguments]

Options:
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
  --ansi               Force ANSI output
  --no-ansi            Disable ANSI output
  -n, --no-interaction Do not ask any interactive question
  -e, --env=ENV        The environment name [default: "dev"]
  --no-debug           Switches off debug mode
  -v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  DataCheck          ...
                    Displays information about the current project
  about             ...
  general           ...
  help              Displays help for a command
  list              Lists commands
  api
  api:doc:dump      Dumps API documentation in various formats
  api:swagger:dump  Dumps Swagger-compliant API definitions.
  assets
  assets:install    Installs bundles web assets under a public directory
  cache
  cache:clear       Clears the cache
  cache:pool:clear  Clears cache pools
  cache:warmup      Warms up an empty cache
  config
  config:dump-reference Dumps the default configuration for an extension
  debug
  debug:config      Dumps the current configuration for an extension
  debug:container   Displays current services for an application
  debug:event-dispatcher Displays configured listeners for an application
  debug:router      Displays current routes for an application
  debug:swiftmailer [swiftmailer:debug] Displays current mailers for an application
  debug:translation Displays translation messages information
  debug:twig        Shows a list of twig functions, filters, globals and tests
  doctrine
  doctrine:cache:clear-collection-region Clear a second-level cache collection region.
  doctrine:cache:clear-entity-region   Clear a second-level cache entity region.
  doctrine:cache:clear-metadata        Clears all metadata cache for an entity manager
  doctrine:cache:clear-query           Clears all query cache for an entity manager
  doctrine:cache:clear-query-region    Clear a second-level cache query region.
  doctrine:cache:clear-result          Clears result cache for an entity manager
  doctrine:cache:contains              Check if a cache entry exists
```

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Σε επόμενο κεφάλαιο θα αναλύσουμε γιατί την συγκεκριμένη εντολή θα την τρέχουμε όπως φαίνεται παρακάτω:

```
docker-compose run --rm --user www-data php bin/console
```

Για να δημιουργήσουμε τις δικές μας εντολές θα πρέπει να δημιουργήσουμε (αν δεν υπάρχει ήδη) στο Bundle που θέλουμε έναν φάκελο με όνομα **Command** και να δημιουργήσουμε ένα αρχείο όπου η κατάληξή του θα είναι **Command.php**.

Παρακάτω φαίνεται η εντολή που δημιουργήθηκε για την αρχικοποίηση μερικών δεδομένων στην βάση δεδομένων για να μπορέσουμε να επιστρέψουμε αποτελέσματα στις κλήσεις του API της διπλωματικής.

Για να εκτελεστεί η εντολή πρέπει να τρέξουμε στο τερματικό μας την παρακάτω εντολή:

```
docker-compose run --rm --user www-data php bin/console DataCheck
```

Έτσι θα δημιουργήσουμε εγγραφές στους αντίστοιχους πίνακες με χρήστες και πληροφορίες για αυτούς, τους λογαριασμούς και τις κάρτες που ανήκουν σε κάθε λογαριασμό αλλά και τέλος μερικές από τις κινήσεις σε αυτούς τους λογαριασμούς.

```
<?php

namespace AppBundle\Command;

use AppBundle\Entity\Account;
use AppBundle\Entity\Bank;
use AppBundle\Entity/Card;
use AppBundle\Entity\Transaction;
use AppBundle\Entity\User;
use AppBundle\Entity\UserInfo;
use Symfony\Bundle\FrameworkBundle\Command\ContainerAwareCommand;
use Symfony\Component\Console\Input\InputArgument;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Input\InputOption;
use Symfony\Component\Console\Output\OutputInterface;

class DataCheckCommand extends ContainerAwareCommand
{
    protected function configure()
    {
        $this
            ->setName('DataCheck')
            ->setDescription('...')
            ->addArgument('argument', InputArgument::OPTIONAL, 'Argument description')
            ->addOption('option', null, InputOption::VALUE_NONE, 'Option description')
        ;
    }

    protected function execute(InputInterface $input, OutputInterface $output)
    {
```

```
$argument = $input->getArgument('argument');

$this->createMembers();
$this->createDummyData();

$output->writeln('Data initialised');
}

protected function createMembers()
{
    $users = [
        'stelios' => [
            'id' => 1,
            'firstname' => 'Στυλιανός',
            'lastname' => 'Μαυρουδής',
            'fathername' => 'Πναγιώτης',
            'address' => 'Καραϊσκου 143',
            'city' => 'Πειραιάς',
            'country' => 'Ελλάδα',
            'postal_code' => '185 35'
        ],
        'george' => [
            'id' => 2,
            'firstname' => 'Γεώργιος',
            'lastname' => 'Παπαχρήστου',
            'fathername' => 'Διογέννης',
            'address' => 'Εφέσσου 6',
            'city' => 'Νέα Σμύρνη',
            'country' => 'Ελλάδα',
            'postal_code' => '171 21'
        ]
    ];

    foreach ($users as $username => $data) {

        $u = new User();
        $u->setId($data['id']);
        $u->setEmail("{ $username}@mail.com");
        $u->setEnabled(1);
        $u->setPlainPassword('123456');
        $u->setUsername($username);

        $ui = new UserInfo();
        $ui->setFatherName($data['fathername']);
        $ui->setFirstName($data['firstname']);
        $ui->setLastName($data['lastname']);
        $ui->setAddress($data['address']);
        $ui->setCity($data['city']);
        $ui->setCountry($data['country']);
        $ui->setPostalCode($data['postal_code']);

        $em = $this->getContainer()->get('doctrine')->getManager();

        try {
            $em->persist($u);
            $em->flush();
        }
    }
}
```

```
        $ui->setUserId($u->getId());
        $ui->setUser($u);

        $em->persist($ui);
        $em->flush();

    } catch (\Exception $e) {
        echo $e->getMessage();
    }
}

protected function createDummyData()
{
    $em = $this->getContainer()->get('doctrine')->getManager();

    $banks = [
        'peiraios' => [
            'id' => 1,
            'name' => 'Παιραιώς',
            'type' => 1,
            'number' => 19,
            'address' => 'Καραϊσκου 143',
            'city' => 'Πειραιάς',
            'country' => 'Ελλάδα',
            'postal_code' => '185 35'
        ],
        'eurobank' => [
            'id' => 2,
            'name' => 'Eurobank',
            'type' => 1,
            'number' => 55,
            'address' => 'Καραϊσκου 143',
            'city' => 'Πειραιάς',
            'country' => 'Ελλάδα',
            'postal_code' => '185 35'
        ],
        'ethniki' => [
            'id' => 3,
            'name' => 'Εθνική',
            'type' => 1,
            'number' => 23,
            'address' => 'Καραϊσκου 143',
            'city' => 'Πειραιάς',
            'country' => 'Ελλάδα',
            'postal_code' => '185 35'
        ]
    ];

    foreach ($banks as $bankRecord => $data) {

        $bank = new Bank();
        $bank->setId($data['id']);
        $bank->setName($data['name']);
        $bank->setType($data['type']);
    }
}
```

```
$bank->setNumber($data['number']);
$bank->setAddress($data['address']);
$bank->setCountry($data['country']);
$bank->setCity($data['country']);
$bank->setPostalCode($data['postal_code']);

try {
    $em->persist($bank);
    $em->flush();
} catch (\Exception $e) {
    echo $e->getMessage();
}
}

$accounts = [
    'account_1' => [
        'id' => 1,
        'user_id' => 1,
        'bank_id' => 1,
        'type' => 1,
        'name' => 'ΠΕΙΡΑΙΩΣ ΑΠΟΔΟΧΩΝ',
        'friendly_name' => 'ΠΕΙΡΑΙΩΣ ΑΠΟΔΟΧΩΝ',
        'iban' => 'GR 54ABNA0256094720',
        'available_balance' => 300.15,
        'accounting_balance' => 300.15,
    ],
    'account_2' =>[
        'id' => 2,
        'user_id' => 1,
        'bank_id' => 2,
        'type' => 2,
        'name' => 'EUROBANK ΤΑΜΙΕΥΤΗΡΙΟΥ',
        'friendly_name' => 'EUROBANK ΤΑΜΙΕΥΤΗΡΙΟΥ',
        'iban' => 'GR 87ABNA0898216472',
        'available_balance' => 2000.15,
        'accounting_balance' => 2000.15,
    ],
    'account_3' => [
        'id' => 3,
        'user_id' => 1,
        'bank_id' => 3,
        'type' => 2,
        'name' => 'ΕΘΝΙΚΗ ΤΑΜΙΕΥΤΗΡΙΟΥ',
        'friendly_name' => 'ΕΘΝΙΚΗ ΤΑΜΙΕΥΤΗΡΙΟΥ',
        'iban' => 'GR 45ABNA0422054025',
        'available_balance' => 1500.15,
        'accounting_balance' => 1500.15,
    ],
    'account_4' => [
        'id' => 4,
        'user_id' => 2,
        'bank_id' => 1,
        'type' => 1,
        'name' => 'ΠΕΙΡΑΙΩΣ ΑΠΟΔΟΧΩΝ',
        'friendly_name' => 'ΠΕΙΡΑΙΩΣ ΑΠΟΔΟΧΩΝ',
        'iban' => 'GR 02RAB00115703365',
```



```
        'available_balance' => 500.15,  
        'accounting_balance' => 500.15,  
    ],  
    'account_5' => [  
        'id' => 5,  
        'user_id' => 2,  
        'bank_id' => 2,  
        'type' => 2,  
        'name' => 'EUROBANK TAMIEYTHPIOY',  
        'friendly_name' => 'EUROBANK TAMIEYTHPIOY',  
        'iban' => 'GR 70INGB0791097101',  
        'available_balance' => 3000.15,  
        'accounting_balance' => 3000.15,  
    ],  
    'account_6' => [  
        'id' => 6,  
        'user_id' => 2,  
        'bank_id' => 3,  
        'type' => 2,  
        'name' => 'ΕΘΝΙΚΗ TAMIEYTHPIOY',  
        'friendly_name' => 'ΕΘΝΙΚΗ TAMIEYTHPIOY',  
        'iban' => 'GR 76INGB0677207159',  
        'available_balance' => 4500.15,  
        'accounting_balance' => 4500.15,  
    ],  
];  
  
foreach ($accounts as $accountRecord => $data) {  
  
    $account = new Account();  
    $account->setId($data['id']);  
    $account->setUserId($data['user_id']);  
    $account->setBankId($data['bank_id']);  
    $account->setNumber(rand(1,10));  
    $account->setType($data['type']);  
    $account->setName($data['name']);  
    $account->setFriendlyName($data['friendly_name']);  
    $account->setIban($data['iban']);  
    $account->setAvailableBalance($data['available_balance']);  
    $account->setAccountingBalance($data['accounting_balance']);  
  
    $owner = $this->getContainer()  
        ->get('doctrine')  
        ->getRepository('AppBundle:User')  
        ->findOneById($data['user_id']);  
  
    $account->setOwner($owner);  
  
    try {  
        $em->persist($account);  
        $em->flush();  
    } catch (\Exception $e) {  
        echo $e->getMessage();  
    }  
}
```

```
$cards = [  
  'card_1' => [  
    'number' => '4111111111111111', # VISA test card  
    'userId' => 1,  
    'type' => 1,  
    'dayLimitAmount' => 600,  
    'buyLimit' => 600,  
    'withdrawalAvailableAmount' => 300,  
    'dateCreated' => new \DateTime('now'),  
    'status' => 1,  
    'expirationDate' => new \DateTime('2021-01-01T15:03:01.012345Z'),  
    'accountId' => 1,  
    'ownerName' => 'MAVROUDIS STYLIANOS'  
  ],  
  'card_2' =>[  
    'number' => '5555555555554444', # MASTER CARD test card  
    'userId' => 1,  
    'type' => 2,  
    'dayLimitAmount' => 600,  
    'buyLimit' => 600,  
    'withdrawalAvailableAmount' => 2000,  
    'dateCreated' => new \DateTime('now'),  
    'status' => 1,  
    'expirationDate' => new \DateTime('now'),  
    'accountId' => 2,  
    'ownerName' => 'MAVROUDIS STYLIANOS'  
  ],  
  'card_3' => [  
    'number' => '401288888881881', # VISA test card  
    'userId' => 2,  
    'type' => 1,  
    'dayLimitAmount' => 600,  
    'buyLimit' => 600,  
    'withdrawalAvailableAmount' => 500,  
    'dateCreated' => new \DateTime('now'),  
    'status' => 1,  
    'expirationDate' => new \DateTime('2021-01-01T15:03:01.012345Z'),  
    'accountId' => 4,  
    'ownerName' => 'PAPACHRISTOU GEORGE'  
  ],  
  'card_4' =>[  
    'number' => '5105105105100', # MASTER CARD test card  
    'userId' => 2,  
    'type' => 2,  
    'dayLimitAmount' => 600,  
    'buyLimit' => 600,  
    'withdrawalAvailableAmount' => 3000,  
    'dateCreated' => new \DateTime('now'),  
    'status' => 1,  
    'expirationDate' => new \DateTime('2021-01-01T15:03:01.012345Z'),  
    'accountId' => 4,  
    'ownerName' => 'PAPACHRISTOU GEORGE'  
  ]  
];
```

```
foreach ($cards as $cardRecord => $data) {

    $card = new Card();
    $card->setNumber($data['number']);
    $card->setUserId($data['userId']);
    $card->setType($data['type']);
    $card->setDayLimitAmount(600);
    $card->setBuyLimit($data['buyLimit']);
    $card->setWithdrawalAvailableAmount($data['withdrawalAvailableAmount']);
    $card->setDateCreated($data['dateCreated']);
    $card->setStatus($data['status']);
    $card->setExpirationDate($data['expirationDate']);
    $card->setAccountId($data['accountId']);
    $card->setOwnerName($data['ownerName']);

    try {
        $em->persist($card);
        $em->flush();
    } catch (\Exception $e) {
        echo $e->getMessage();
    }
}

$transactions = [
    [
        'accountId' => 1,
        'cardId' => 1,
        'type' => 1,
        'amount' => -48.5,
        'status' => 'P',
        'info' => 'ΠΛΗΡΩΜΗ ΔΕΗ 713325977XXX WB'
    ],
    [
        'accountId' => 1,
        'cardId' => 0,
        'type' => 2,
        'amount' => 700,
        'status' => 'A',
        'info' => 'ΜΙΣΘΟΔΟΣΙΑ'
    ],
    [
        'accountId' => 1,
        'cardId' => 0,
        'type' => 3,
        'amount' => 350,
        'status' => 'A',
        'info' => 'ΠΛΗΡΩΜΗ ΕΝΟΙΚΙΟΥ'
    ],
    [
        'accountId' => 4,
        'cardId' => 0,
        'type' => 1,
        'amount' => -50,
        'status' => 'A',
        'info' => 'ΠΛΗΡΩΜΗ ΔΕΗ 733325977XXX WB'
    ],
],
```

```
[
    [
        'accountId' => 4,
        'cardId' => 0,
        'type' => 2,
        'amount' => 700,
        'status' => 'A',
        'info' => 'ΜΙΣΘΟΔΟΣΙΑ'
    ],
    [
        'accountId' => 4,
        'cardId' => 0,
        'type' => 3,
        'amount' => 350,
        'status' => 'A',
        'info' => 'ΠΛΗΡΩΜΗ ΕΝΟΙΚΙΟΥ'
    ]
];

foreach ($transactions as $data) {
    $tr = new Transaction();

    $tr->setAccountId($data['accountId']);
    $tr->setStatus($data['status']);
    $tr->setType($data['type']);
    $tr->setCardId($data['cardId']);
    $tr->setInfo($data['info']);
    $tr->setAmount($data['amount']);

    try {
        $em->persist($tr);
        $em->flush();
    } catch (\Exception $e) {
        echo $e->getMessage();
    }
}
}
```

Για να δημιουργήσουμε μια δική μας εντολή εναλλακτικά μπορούμε να τρέξουμε την εντολή που ακολουθεί και να συνεχίσουμε βήμα βήμα:

```
docker-compose run --rm --user www-data php bin/console generate:command
```

Παρακάτω περιγράφονται μερικές από τις πιο βασικές εντολές του Symfony:

Δημιουργία του σχήματος της βάσης δεδομένων.

```
php bin/console doctrine:database:create
```

Εφαρμογή αλλαγών στην δομή της βάσης δεδομένων.

```
php bin/console doctrine:schema:update
```

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Υπεύθυνο για τα μοντέλα και την λειτουργικότητα που αφορούν τον χρήστη (web user) της εφαρμογής είναι το FOS User Bundle (FOS: Friends of symfony). Έτσι με τις παρακάτω εντολές έχουμε διάφορες λειτουργίες για τους χρήστες μέσω του τερματικού μας.

Ενεργοποίηση ενός χρήστη.

```
php bin/console fos:user:activate
```

Αλλαγή κωδικού πρόσβασης για έναν χρήστη.

```
php bin/console fos:user:change-password
```

Δημιουργία νέου χρήστη.

```
php bin/console fos:user:create
```

Απενεργοποίηση ενός χρήστη. Ο χρήστης δεν θα μπορεί να εισέλθει στην εφαρμογή μας.

```
php bin/console fos:user:deactivate
```

Δημιουργία ενός νέου Bundle.

```
php bin/console generate:bundle
```

Δημιουργία ενός νέου Controller

```
php bin/console generate:controller
```

Έλεγχος της ασφάλειας της εφαρμογής.

```
php bin/console security:check
```

2.9 Αυθεντικοποίηση χρήστη μέσω του API

Για να μπορέσουμε να κάνουμε το API μας ασφαλέστερο αλλά και για να μπορέσουμε να κάνουμε αυθεντικοποίηση χρηστών σε αυτό θα πρέπει να εγκαταστήσουμε και να παραμετροποιήσουμε την βιβλιοθήκη **lexik/jwt-authentication-bundle**. [13]

Εγκατάσταση της βιβλιοθήκης.

```
docker-compose run --rm composer require lexik/jwt-authentication-bundle
```

Αφού τελειώσει η εγκατάσταση της βιβλιοθήκης θα πρέπει να την ενεργοποιήσουμε προσθέτοντας την παρακάτω γραμμή στο αρχείο **AppKernel.php** στην μέθοδο **registerBundles()**.

Συνολικά το αρχείο **AppKernel.php** θα πρέπει να είναι:

```
<?php

use Symfony\Component\HttpKernel\Kernel;
use Symfony\Component\Config\Loader\LoaderInterface;

class AppKernel extends Kernel
{
    public function registerBundles()
    {
        $bundles = [
            new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
            new Symfony\Bundle\SecurityBundle\SecurityBundle(),
            new Symfony\Bundle\TwigBundle\TwigBundle(),
            new Symfony\Bundle\MonologBundle\MonologBundle(),
            new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),
            new Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),
            new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),
            new AppBundle\AppBundle(),
            new FOS\UserBundle\FOSUserBundle(),
            new FOS\RestBundle\FOSRestBundle(),
            new JMS\SerializerBundle\JMSSerializerBundle(),
            new Nelmio\CorsBundle\NelmioCorsBundle(),
            new Nelmio\ApiDocBundle\NelmioApiDocBundle(),
            new ApiBundle\ApiBundle(),
            new \Lexik\Bundle\JWTAuthenticationBundle\LexikJWTAuthenticationBundle()
        ];

        if (in_array($this->getEnvironment(), ['dev', 'test'], true)) {
            $bundles[] = new Symfony\Bundle\DebugBundle\DebugBundle();
            $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();
            $bundles[] = new Sensio\Bundle\DistributionBundle\SensioDistributionBundle();

            if ('dev' === $this->getEnvironment()) {
                $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();
                $bundles[] = new Symfony\Bundle\WebServerBundle\WebServerBundle();
            }
        }
    }
}
```

```
        return $bundles;
    }

    public function getRootDir()
    {
        return __DIR__;
    }

    public function getCacheDir()
    {
        return dirname(__DIR__).'/var/cache/'. $this->getEnvironment();
    }

    public function getLogDir()
    {
        return dirname(__DIR__).'/var/logs';
    }

    public function registerContainerConfiguration(LoaderInterface $loader)
    {
        $loader->load($this->getRootDir().'/config/config_'. $this->getEnvironment().'.yml');
    }
}
```

Στην συνέχεια θα πρέπει να παραμετροποιήσουμε τις τιμές που αφορούν το **lexik/jwt-authentication-bundle** οι οποίες βρίσκονται στο αρχείο **config.yml**.

```
lexik_jwt_authentication:
  private_key_path: '%jwt_private_key_path%'
  public_key_path:  '%jwt_public_key_path%'
  pass_phrase:     '%jwt_key_pass_phrase%'
  token_ttl:       '%jwt_token_ttl%'
```

Για τις τιμές των παραπάνω παραμέτρων πρέπει να ορίσουμε τις αντίστοιχες μεταβλητές στο αρχείο **parameters.yml**.

```
jwt_private_key_path: '%kernel.root_dir%/../var/jwt/private.pem'
jwt_public_key_path:  '%kernel.root_dir%/../var/jwt/public.pem'
jwt_key_pass_phrase:  tei_secure
jwt_token_ttl:        3600
```

Στις μεταβλητές αυτές ορίζουμε την διαδρομή των αρχείων κλειδιών που χρησιμοποιεί το JWT για την κρυπτογράφηση και την αποκρυπτογράφηση ενός κειμένου μαζί με την κωδική φράση που χρειάζεται για να παρέχει παραπάνω ασφάλεια. Με αυτά θα παράγεται ένα κρυπτογραφημένο αλφαριθμητικό (token) κατά την επιτυχημένη είσοδο ενός χρήστη στην εφαρμογή μας μέσω του API και στην συνέχεια με αυτό το token ο χρήστης θα κάνει κλήσεις προς το API.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Το API θα αποκρυπτογραφεί αυτό το token και θα ελέγχει αν είναι ορθό για να αφήσει την πρόσβαση στον χρήστη.

Η παράμετρος **jwt_token_ttl** δείχνει την διάρκεια ζωής ενός token.

Για να δημιουργήσουμε τα παραπάνω αρχεία θα πρέπει να τρέξουμε τις παρακάτω εντολές:

```
mkdir -p var/jwt
openssl genrsa -out var/jwt/private.pem -aes256 4096
openssl rsa -pubout -in var/jwt/private.pem -out var/jwt/public.pem
```

Και στη δημιουργία αρχείων θα μας ζητηθεί να βάλουμε το **passphrase**, όπου στην περίπτωση μας είναι το **tei_secure** όπως φαίνεται και παραπάνω.

```
→ tei_pir_ptuxiaki_api git:(master) X cd app
→ app git:(master) X openssl genrsa -out var/jwt/private.pem -aes256 4096
Generating RSA private key, 4096 bit long modulus
.....++
.....++
e is 65537 (0x010001)
Enter pass phrase for var/jwt/private.pem:
Verifying - Enter pass phrase for var/jwt/private.pem:
→ app git:(master) X openssl rsa -pubout -in var/jwt/private.pem -out var/jwt/public.pem

Enter pass phrase for var/jwt/private.pem:
writing RSA key
```

Αφού έχουμε κάνει όλες τις παραπάνω προεργασίες θα πρέπει να δημιουργήσουμε μια κλάση ώστε να βάλουμε την λογική ώστε να παραχθεί το ζητούμενο **token**. Η κλάση που θα δημιουργήσουμε είναι η **TokenAuthenticator** όπου κληρονομεί την abstract κλάση **AbstractGuardAuthenticator**.

Η συγκεκριμένη κλάση θα χρησιμοποιηθεί από το ApiBundle και συνεπώς θα βρίσκεται στην ακόλουθη διαδρομή **ApiBundle\Security\TokenAuthenticator** για την οποία θα πρέπει να ενημερώσουμε και το αρχείο με τις ρυθμίσεις ασφάλειας του **Symfony (security.yml)**.

Στο security.yml αρχείο στην ενότητα firewalls θα ορίσουμε τους κανόνες που αφορούν το API της εφαρμογής (δηλαδή για για την διαδρομή /api*) όπως φαίνονται παρακάτω:

```
firewalls:
  secured_api:
    pattern: ^/api/
    anonymous: true
    stateless: true
    guard:
      authenticators:
        - ApiBundle\Security\TokenAuthenticator
```


Συνολικά το **security.yml** αρχείο είναι:

```
security:
  encoders:
    FOS\UserBundle\Model\UserInterface: bcrypt

  role_hierarchy:
    ROLE_ADMIN:       ROLE_USER
    ROLE_SUPER_ADMIN: ROLE_ADMIN

  providers:
    database:
      entity:
        class: AppBundle\User
        property: username
    fos_userbundle:
      id: fos_user.user_provider.username

  firewalls:
    secured_api:
      pattern: ^/api/
      anonymous: true
      stateless: true
      guard:
        authenticators:
          - AppBundle\Security\TokenAuthenticator
    main:
      pattern: ^/
      form_login:
        provider: fos_userbundle
        csrf_token_generator: security.csrf.token_manager
        # if you are using Symfony < 2.8, use the following config instead:
        # csrf_provider: form.csrf_provider

      logout: true
      anonymous: true

  access_control:
    - { path: ^/login$, role: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/register, role: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/resetting, role: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/admin/, role: ROLE_ADMIN }
```

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Στην κλάση **TokenAuthenticator** θα ορίσουμε με ποιόν τρόπο θα ελέγχουμε το token της αίτησης αυθεντικοποίησης αλλά και με ποιόν τρόπο θα βρίσκουμε και θα γυρνάμε τον κατάλληλο χρήστη στο σύστημα.

Υπεύθυνες για τα δυο παραπάνω είναι οι μέθοδοι **getUser** και **getCredentials** και θα τις αλλάζουμε όπως φαίνονται παρακάτω:

```
/**
 * Get the authentication credentials from the request and return them
 * as any type (e.g. an associate array). If you return null, authentication
 * will be skipped.
 *
 * Whatever value you return here will be passed to getUser() and checkCredentials()
 *
 * For example, for a form login, you might:
 *
 *     if ($request->request->has('_username')) {
 *         return array(
 *             'username' => $request->request->get('_username'),
 *             'password' => $request->request->get('_password'),
 *         );
 *     } else {
 *         return;
 *     }
 *
 * Or for an API token that's on a header, you might use:
 *
 *     return array('api_key' => $request->headers->get('X-API-TOKEN'));
 *
 * @param Request $request
 *
 * @return mixed|null
 */
public function getCredentials(Request $request)
{
    $extractor = new AuthorizationHeaderTokenExtractor('Bearer', 'Authorization');
    $token = $extractor->extract($request);

    if (!$token) {
        return null;
    }

    return $token;
}
```

Η **getCredentials** παίρνει τον περιεχόμενο του **Authorization Header** με πρόθεμα **Bearer** και στην συνέχεια αποκρυπτογραφεί (λειτουργία `extract`), με τη βοήθεια των κλειδιών που δημιουργήσαμε σε προηγούμενο βήμα, τα στοιχεία του χρήστη (στην περίπτωση μας `username` και `password`). [13]

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Στην συνέχεια αν το token είναι σωστό η getUser το χρησιμοποιεί για να γυρίσει το σωστό μοντέλο χρήστη από την βάση.

```
/**
 * Return a UserInterface object based on the credentials.
 *
 * The *credentials* are the return value from getCredentials()
 *
 * You may throw an AuthenticationException if you wish. If you return
 * null, then a UsernameNotFoundException is thrown for you.
 *
 * @param mixed $credentials
 * @param UserProviderInterface $userProvider
 *
 * @throws AuthenticationException
 *
 * @return UserInterface|null
 */
public function getUser($credentials, UserProviderInterface $userProvider)
{
    try {

        $data = $this->jwtEncoder->decode($credentials);

        if (false === $data) {
            return null;
        }

        return $userProvider->loadUserByUsername($data['username']);
    } catch (JWTDecodeFailureException $exception) {
        return null;
    }
}
```

Στην συνέχεια φαίνεται πώς δημιουργούμε το endpoint του API για την χρήση των παραπάνω με αποτέλεσμα την αυθεντικοποίηση του χρήστη στην εφαρμογή μας μέσω του API.

Στον UsersController δημιουργούμε την μέθοδο (Action method) **postLoginAction**.

```
/**
 * @Rest\Post("/Login")
 */
public function postLoginAction(Request $request)
{
    $username = $request->getUser();
    $password = $request->getPassword();

    $user = $this->getDoctrine()
        ->getRepository('AppBundle:User')
        ->findOneBy(['username' => $username]);

    if (!$user) {
```

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

```
        throw new BadCredentialsException();
    }

    $isPasswordValid = $this->passwordEncoder->isPasswordValid($user, $password);

    if (!$isPasswordValid) {
        throw new BadCredentialsException();
    }

    $token = $this->jwtEncoder->encode(
        [
            'username' => $user->getUsername(),
            'exp'       => time() + 36000,
        ]
    );

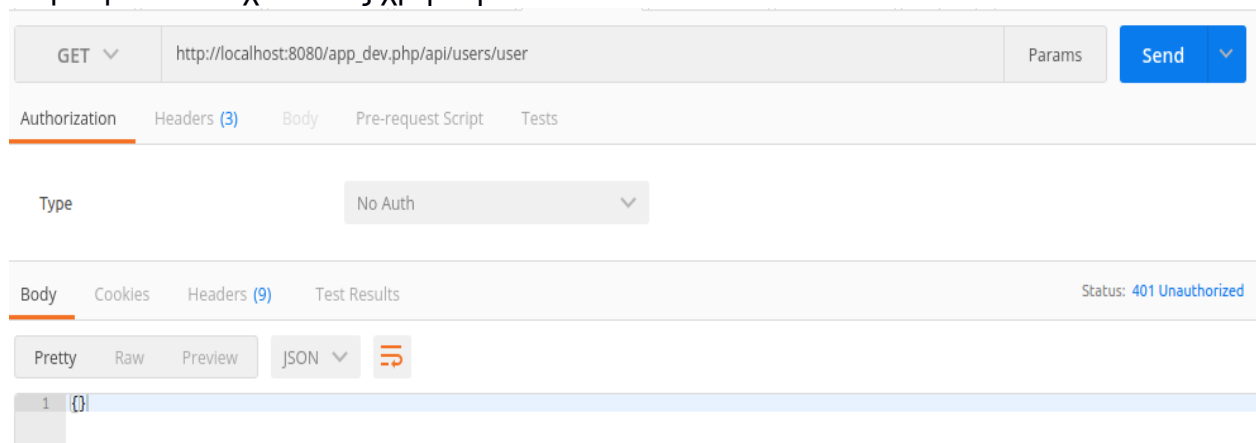
    return new JsonResponse(['token' => $token]);
}
```

Η διαδρομή (endpoint) για την παραπάνω μέθοδο είναι η http://localhost:8080/app_dev.php/api/users/login και υποστηρίζει POST Request.

Στην συνέχεια θα χρησιμοποιήσουμε έναν υπάρχων χρήστη στην βάση δεδομένων μας για να πραγματοποιήσουμε την είσοδο του στο σύστημα μέσω του API με την βοήθεια του εργαλείου Postman.

Το Postman αποτελεί εργαλείο REST Client και μπορούμε εύκολα να προσομοιώνουμε αιτήσεις σε μία διαδικτυακή εφαρμογή.

Αρχικά θα δοκιμάσουμε μια αίτηση (GET Request) στην διαδρομή http://localhost:8080/app_dev.php/api/users/user API χωρίς να έχει γίνει η διαδικασία που περιγράφεται παραπάνω για να δούμε αν θα μπορούσαμε να πάρουμε τα στοιχεία ενός χρήστη.



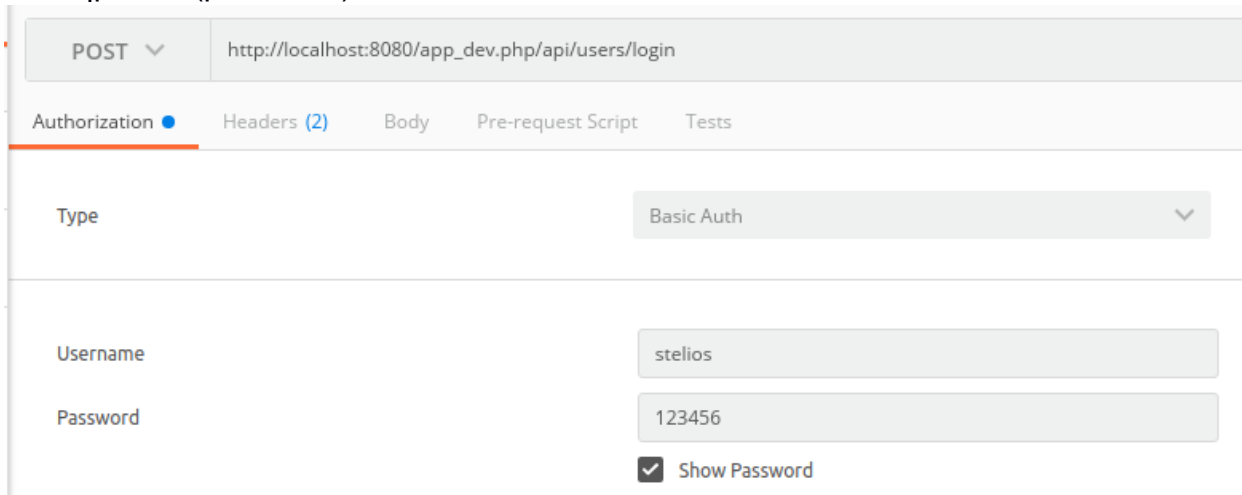
Εικόνα 5 Χρήση εφαρμογής Postman για την αποστολή αιτήσεων στην εφαρμογή

Το αποτέλεσμα είναι κενό όπως περιμέναμε. Θα δοκιμάσουμε το ίδιο αφού πρώτα πραγματοποιήσουμε μια POST αίτηση (POST Request) στην διαδρομή (endpoint)

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

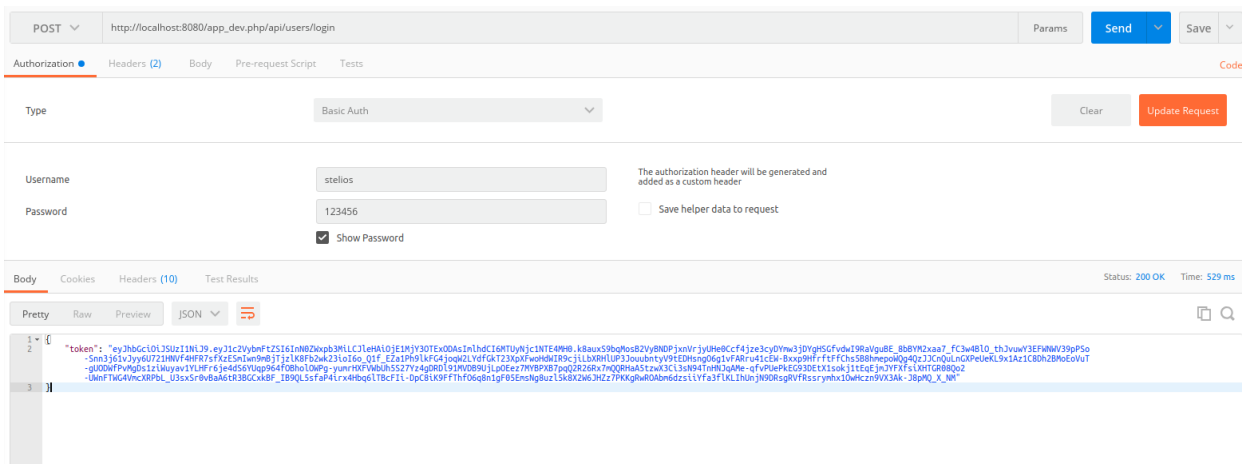
http://localhost:8080/app_dev.php/api/users/login στέλνοντας μαζί και τα συνθηματικά του χρήστη.

Για το παράδειγμά μας το όνομα χρήστη (username) είναι **stelios** και το συνθηματικό (password) είναι το **123456** :



Εικόνα 6 Αυθεντικοποίηση χρήστη μέσω API

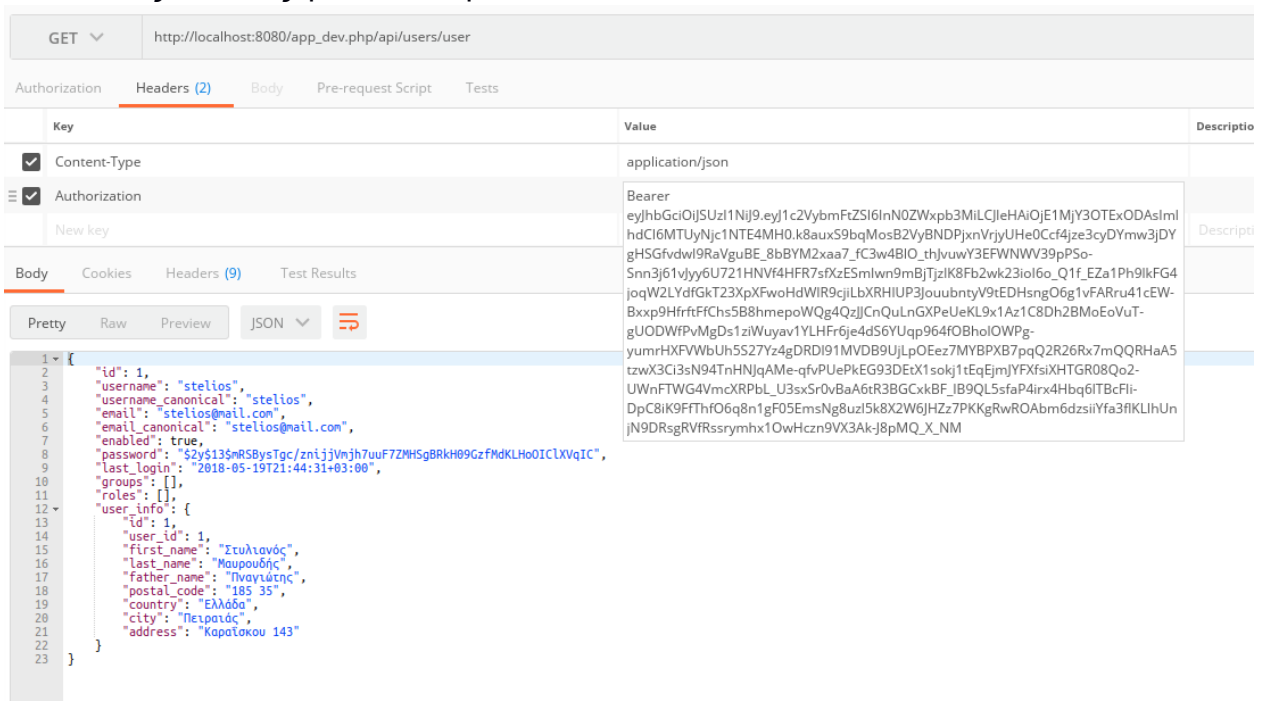
Εφόσον τα συνθηματικά του χρήστη είναι σωστά το API θα δημιουργήσει και θα γυρίσει ως αποτέλεσμα ένα ορθό **token** που θα το χρησιμοποιούμε σαν κλειδί για την χρήση του API. [13]



Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

```
{
  "token": "eyJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6InN0ZWxpY3MiLCJleHAiOjE1MjY3OTExODAsIm1hdCI6MTUyNjc1NTE4MH0.k8auxS9bqMosB2VyBNDPjxnVrjyUHe0Ccf4jze3cyDYmw3jDYgHSGfvdwI9R
  Snn3j61vJyy6U721HNVf4HFR7sfXzESmIwn9mBjTjz1K8Fb2wk23ioI6o_Q1f_EZa1Ph9lkFG4joqW2LYdf
  W-Bxxp9HfrftFfChs5B8hmepoWQg4QzJJCnQuLnGXPeUeKL9x1Az1C8Dh2BMoEoVuT-gUODWfPvMgDs1zi
  yumrHXFVWbUh5S27Yz4gDRD191MVD9UjLpOEz7MYBPXB7pqQ2R26Rx7mQQRHaA5tzW3Ci3sN94TnHNJq
  UWnFTWG4VmcXRPbL_U3sXr0vBaA6tR3BGCxkBF_IB9QL5sfaP4irx4Hbq6lTBcFIi-
  DpC8iK9FfThfO6q8n1gF05EmsNg8uz15k8X2W6JHZz7PKKgRwROAbm6dzsiiYfa3f1KLlHUnjN9DRsgRVfR
  }
}
```

Δοκιμάζουμε πάλι το πάρουμε τα στοιχεία του χρήστη χρησιμοποιώντας το **token** στέλνοντας το όπως φαίνεται παρακάτω:



The screenshot shows a REST client interface with a GET request to `http://localhost:8080/app_dev.php/api/users/user`. The response body is a JSON object with the following structure:

```
{
  "id": 1,
  "username": "stelios",
  "username_canonical": "stelios",
  "email": "stelios@mail.com",
  "email_canonical": "stelios@mail.com",
  "enabled": true,
  "password": "$2y$13$mRSyTgc/zniJJVnjh7uuF7ZMHsgBRkH09GzFmDKLHo0IC1XVqIC",
  "last_login": "2018-05-19T21:44:31+03:00",
  "groups": [],
  "roles": [],
  "user_info": {
    "id": 1,
    "user_id": 1,
    "first_name": "Στυλιανός",
    "last_name": "Μαρουδής",
    "father_name": "Παναγιώτης",
    "postal_code": "185 35",
    "country": "Ελλάδα",
    "city": "Πειραιάς",
    "address": "Καραϊσκού 143"
  }
}
```

Εικόνα 7 Αίτηση στοιχείων χρήστη από το API

Αυτό γιατί η μέθοδος **getCredentials** στην **TokenAuthenticator** κλάση ελέγχει από την επικεφαλίδα της αίτησης (Header Request) την τιμή του πεδίου **Authorization** που ξεκινάει με **Bearer**:

```
$extractor = new AuthorizationHeaderTokenExtractor('Bearer', 'Authorization');
$token = $extractor->extract($request);
```

Έτσι έχουμε τα στοιχεία του χρήστη μας ως αποτέλεσμα:

```
{
  "id": 1,
  "username": "stelios",
  "username_canonical": "stelios",
  "email": "stelios@mail.com",
  "email_canonical": "stelios@mail.com",
```

```
"enabled": true,  
"password":  
"$2y$13$mRSBysTgc/znijjVmjh7uuF7ZMHSgBRkH09GzfMdKLHoOIClXVqIC",  
"last_login": "2018-05-19T21:44:31+03:00",  
"groups": [],  
"roles": [],  
"user_info": {  
  "id": 1,  
  "user_id": 1,  
  "first_name": "Στυλιανός",  
  "last_name": "Μαυρουδής",  
  "father_name": "Παναγιώτης",  
  "postal_code": "185 35",  
  "country": "Ελλάδα",  
  "city": "Πειραιάς",  
  "address": "Καραϊσκού 143"  
}  
}
```

ΚΕΦΑΛΑΙΟ 3

Λειτουργία Διαδικτυακής Εφαρμογής

3.1 Περιγραφή της λειτουργίας του API της εφαρμογής

Η βασική λειτουργία του API είναι να λειτουργεί σαν γέφυρα μεταξύ ενός mobile application με τις διαφορετικές τράπεζες στις οποίες μπορεί να έχει λογαριασμό κάποιος με σκοπό την παροχή συγκεντρωτικής/συνολικής πληροφορίας.

Ένας χρήστης μπορεί να έχει πολλούς διαφορετικούς τραπεζικούς λογαριασμούς σε διαφορετικές τράπεζες.

Σύνδεσμος : http://localhost:8080/app_dev.php/api/accounts/list

```
[
  {
    "id": 1,
    "user_id": 1,
    "name": "ΠΕΙΡΑΙΩΣ ΑΠΟΔΟΧΩΝ",
    "type": 1,
    "bank_id": 1,
    "number": 5,
    "iban": "GR 54ABNA0256094720",
    "available_balance": 300.15,
    "accounting_balance": 300.15,
    "friendly_name": "ΠΕΙΡΑΙΩΣ ΑΠΟΔΟΧΩΝ",
    "owner": {
      "id": 1,
      "username": "stelios",
      "username_canonical": "stelios",
      "email": "stelios@mail.com",
      "email_canonical": "stelios@mail.com",
      "enabled": true,
      "password":
"$2y$13$mRSBysTgc/znijjVmjh7uuF7ZMHSgBRkH09GzfMdKlHoOIClXVqIC",
      "last_login": "2018-05-20T00:21:57+03:00",
      "groups": [],
      "roles": [],
      "user_info": {
        "id": 1,
        "user_id": 1,
        "first_name": "Στυλιανός",
        "last_name": "Μαυρουδής",
        "father_name": "Παναγιώτης",
```



```
        "postal_code": "185 35",
        "country": "Ελλάδα",
        "city": "Πειραιάς",
        "address": "Καραϊσκού 143"
    }
}
},
{
    "id": 2,
    "user_id": 1,
    "name": "EUROBANK TAMIEYTHPIOY",
    "type": 2,
    "bank_id": 2,
    "number": 7,
    "iban": "GR 87ABNA0898216472",
    "available_balance": 2000.15,
    "accounting_balance": 2000.15,
    "friendly_name": "EUROBANK TAMIEYTHPIOY",
    "owner": {
        "id": 1,
        "username": "stelios",
        "username_canonical": "stelios",
        "email": "stelios@mail.com",
        "email_canonical": "stelios@mail.com",
        "enabled": true,
        "password":
"$2y$13$mRSBysTgc/znijjVmjh7uuF7ZMHSgBRkH09GzfMdKLHoOIC1XVqIC",
        "last_login": "2018-05-20T00:21:57+03:00",
        "groups": [],
        "roles": [],
        "user_info": {
            "id": 1,
            "user_id": 1,
            "first_name": "Στυλιανός",
            "last_name": "Μαυρουδής",
            "father_name": "Παναγιώτης",
            "postal_code": "185 35",
            "country": "Ελλάδα",
            "city": "Πειραιάς",
            "address": "Καραϊσκού 143"
        }
    }
}
```

```
},
{
  "id": 3,
  "user_id": 1,
  "name": "ΕΘΝΙΚΗ ΤΑΜΙΕΥΤΗΡΙΟΥ",
  "type": 2,
  "bank_id": 3,
  "number": 2,
  "iban": "GR 45ABNA0422054025",
  "available_balance": 1500.15,
  "accounting_balance": 1500.15,
  "friendly_name": "ΕΘΝΙΚΗ ΤΑΜΙΕΥΤΗΡΙΟΥ",
  "owner": {
    "id": 1,
    "username": "stelios",
    "username_canonical": "stelios",
    "email": "stelios@mail.com",
    "email_canonical": "stelios@mail.com",
    "enabled": true,
    "password":
"$2y$13$mRSBySTgc/znijjVmjh7uuF7ZMHSgBRkH09GzfMdKLHoOIC1XVqIC",
    "last_login": "2018-05-20T00:21:57+03:00",
    "groups": [],
    "roles": [],
    "user_info": {
      "id": 1,
      "user_id": 1,
      "first_name": "Στυλιανός",
      "last_name": "Μαυρουδής",
      "father_name": "Παναγιώτης",
      "postal_code": "185 35",
      "country": "Ελλάδα",
      "city": "Πειραιάς",
      "address": "Καραϊσκου 143"
    }
  }
}
]
```

Ένας χρήστης μπορεί να έχει πολλές διαφορετικές κάρτες συνδεδεμένες με τους τραπεζικούς λογαριασμούς.

Σύνδεσμος: http://localhost:8080/app_dev.php/api/cards/list

```
[
  {
    "id": 1,
    "number": "4111111111111111",
    "user_id": 1,
    "type": 1,
    "day_limit_amount": 600,
    "buy_limit": 600,
    "withdrawal_available_amount": 300,
    "date_created": "2018-04-29T00:00:00+03:00",
    "status": 1,
    "expiration_date": "2021-01-01T00:00:00+02:00",
    "account_id": 1,
    "owner_name": "MAVROUDIS STYLIANOS"
  },
  {
    "id": 2,
    "number": "5555555555554444",
    "user_id": 1,
    "type": 2,
    "day_limit_amount": 600,
    "buy_limit": 600,
    "withdrawal_available_amount": 2000,
    "date_created": "2018-04-29T00:00:00+03:00",
    "status": 1,
    "expiration_date": "2018-04-29T00:00:00+03:00",
    "account_id": 2,
    "owner_name": "MAVROUDIS STYLIANOS"
  }
]
```

Ένας χρήστης μπορεί να δει το συνολικό χρηματικό υπόλοιπο από όλους τους τραπεζικούς λογαριασμούς .

Σύνδεσμος : http://localhost:8080/app_dev.php/api/users/totalBalance

```
{
  "total_balance": "3800.4500000000003"
}
```

Ένας χρήστης μπορεί να δει αναλυτικά τις κινήσεις ενός τραπεζικού λογαριασμού.

Σύνδεσμος : http://localhost:8080/app_dev.php/api/transactions/list/1

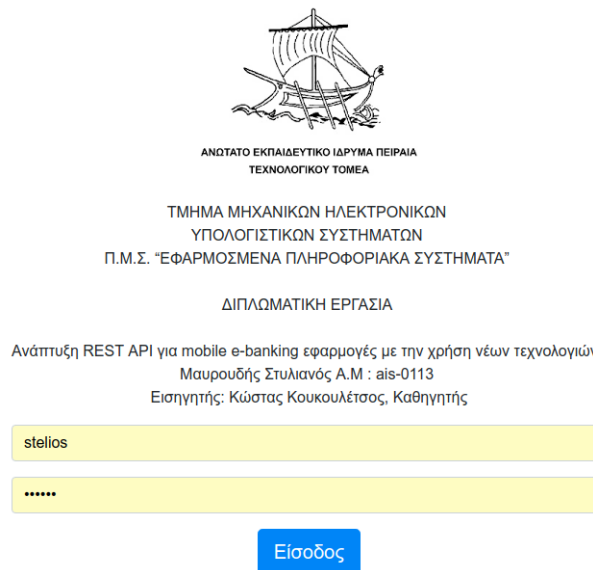
```
[
  {
    "id": 1,
    "account_id": 1,
    "card_id": 1,
    "type": 1,
    "amount": -48.5,
    "info": "ΠΛΗΡΩΜΗ ΔΕΗ 713325977XXX WB",
    "status": "P"
  },
  {
    "id": 2,
    "account_id": 1,
    "card_id": 0,
    "type": 2,
    "amount": 700,
    "info": "ΜΙΣΘΟΔΟΣΙΑ",
    "status": "A"
  },
  {
    "id": 3,
    "account_id": 1,
    "card_id": 0,
    "type": 3,
    "amount": 350,
    "info": "ΠΛΗΡΩΜΗ ΕΝΟΙΚΙΟΥ",
    "status": "A"
  }
]
```

3.2 Περιγραφή της λειτουργίας της Δικτυακής εφαρμογής

Το AppBundle που δημιουργήσαμε σε προηγμένη ενότητα περιέχει τον Controller που είναι υπεύθυνος να σερβίρει τις σελίδες της εφαρμογής στον χρήστη με αντίστοιχες λειτουργίες του API.

1. Είσοδος του χρήστη στην διαδικτυακή εφαρμογή.

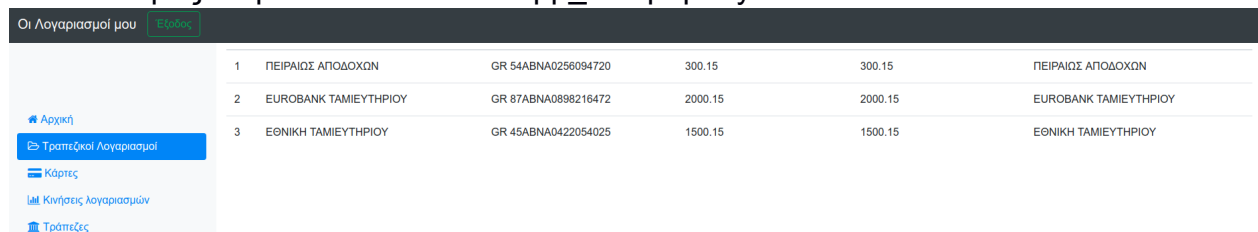
Σύνδεσμος: http://localhost:8080/app_dev.php/login



Εικόνα 8 Εισαγωγή χρήστη μέσω της διαδικτυακής εφαρμογής

2. Λίστα τραπεζικών λογαριασμών χρήστη

Σύνδεσμος: http://localhost:8080/app_dev.php/myAccounts



Α/Α	Όνομα Τράπεζας	Αριθμός Λογαριασμού	Ποσό	Ποσό	Όνομα Τράπεζας
1	ΠΕΙΡΑΙΩΣ ΑΠΟΔΟΧΩΝ	GR 54ABNA0256094720	300.15	300.15	ΠΕΙΡΑΙΩΣ ΑΠΟΔΟΧΩΝ
2	EUROBANK ΤΑΜΙΕΥΤΗΡΙΟΥ	GR 87ABNA0898216472	2000.15	2000.15	EUROBANK ΤΑΜΙΕΥΤΗΡΙΟΥ
3	ΕΘΝΙΚΗ ΤΑΜΙΕΥΤΗΡΙΟΥ	GR 45ABNA0422054025	1500.15	1500.15	ΕΘΝΙΚΗ ΤΑΜΙΕΥΤΗΡΙΟΥ

Εικόνα 9 Λίστα τραπεζικών λογαριασμών μέσω δικτυακής εφαρμογής

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

3. Λίστα καρτών χρήστη

Σύνδεσμος: http://localhost:8080/app_dev.php/myCards

#	Αριθμός	Ημερήσιο Όριο Αγορών	Αγορές Εντός και Εκτός Ελλάδος	Όριο Αναλήψεων	Όνομα Κατόχου	Ημ/νία Έκδοσης	Ημ/νία Λήξης
1	4111111111111111	600	600	300	MAVROUDIS STYLIANOS	04/29/2018	01/01/2021
2	5555555555554444	600	600	2000	MAVROUDIS STYLIANOS	04/29/2018	04/29/2018

Εικόνα 10 Λίστα καρτών χρήστη μέσω διαδικτυακής εφαρμογής

4. Λίστα κινήσεων λογαριασμού ενός χρήστη.

Σύνδεσμος: http://localhost:8080/app_dev.php/myTransactions

#	Ποσό	Περιγραφή Συναλλαγής	Τραπεζικός Λογαριασμός	Κατάσταση
1	-48.5	ΠΛΗΡΩΜΗ ΔΕΗ 713325977XXX WB	1	Εκκρεμεί
2	700	ΜΙΣΘΟΔΟΣΙΑ	1	Εκτελεσμένη
3	350	ΠΛΗΡΩΜΗ ΕΝΟΙΚΙΟΥ	1	Εκκρεμεί

Εικόνα 11 Κινήσεις λογαριασμών χρήστη μέσω διαδικτυακής εφαρμογής

5. Λίστα τραπεζών

Σύνδεσμος: http://localhost:8080/app_dev.php/banks

#	Όνομα	Αριθμός Καταστήματος	Ταχυδρομικός Κώδικας	Χώρα	Διεύθυνση
1	Παιραιώς	19	185 35	Ελλάδα	Καρσίσκου 143
2	Eurobank	55	185 35	Ελλάδα	Καρσίσκου 143
3	Εθνική	23	185 35	Ελλάδα	Καρσίσκου 143

Εικόνα 12 Λίστα τραπεζών

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

ΚΕΦΑΛΑΙΟ 4

Cloud Computing και εικονοποίηση

4.1 Περιγραφή του Cloud Computing

Η μετάβαση από διακομιστές σε ιδιωτικά κέντρα δεδομένων σε παροχείς υπηρεσιών Cloud ήταν δραματική. Οι παροχείς υπηρεσιών Cloud δημιουργούν μια τεχνικά προηγμένη υποδομή που οι περισσότερες επιχειρήσεις δεν μπορούν να καλύψουν, βρίσκουν τα κέντρα δεδομένων τους σε περιοχές με φθηνή ηλεκτρική ενέργεια και με υψηλές ταχύτητες δικτύου. Σχεδιάζουν και προσαρμόζουν το πλαίσιο των διακομιστών έτσι ώστε να μεγιστοποιούν την ενεργειακή απόδοση και ελαχιστοποιούν τη ανάγκη συντήρησης. Χρησιμοποιούν ειδικά σχεδιασμένη υποδομή δικτύου με προσαρμοσμένο υλικό και λογισμικό ρυθμισμένο με ακρίβεια στα εσωτερικά τους δίκτυα. Αυτοματοποιούν διαδικασίες ώστε να επιτρέπουν την ταχεία επέκταση και να μειώσει την πιθανότητα ανθρώπινου λάθους. [3]

Ένα επίπεδο πάνω σε αυτό το υπόβαθρο υλικού είναι οι τρόπος διαχείρισης τους που απλοποιούν και διευκολύνουν τη διαμόρφωση της υποδομής. Οι παροχείς υπηρεσιών Cloud παρέχουν τόσο API όσο και εργαλεία που αντιμετωπίζουν οι χρήστες ελέγχουν την παροχή και την αποδέσμευση των πόρων. Ως αποτέλεσμα, ολόκληρος ο κύκλος ζωής ενός συστήματος ή ομάδα συστημάτων που διανέμονται σε ένα εικονικό δίκτυο, μπορούν να αυτοματοποιηθούν. Η ιδέα αυτή περνάει το όνομα "υποδομή ως κώδικας" (Infrastructure As Code), και έρχεται σε αντίθεση με τις διαδικασίες παροχής των παρελθόντων χρόνων.

Η ελαστικότητα είναι ακόμα ένας λόγος για να μεταβεί κάποιος σε υπηρεσίες Cloud αφού ανά πάσα στιγμή μπορεί να ενεργοποιήσει κάποιος ή να απενεργοποιήσει μια υπηρεσία ανάλογα με την ανάγκες. Η εύκολη κλιμάκωση αποτελεί μεγάλο προνόμιο των υπηρεσιών Cloud , αφού το μέγεθος της δύναμης της υποδομής μπορεί να αλλάξει ανάλογα με το φορτίο που μπορεί να έχει την κάθε στιγμή το σύστημα και να την αυξομειώνει δυναμικά.

Έτσι βελτιώνονται ακόμα και τα κόστη, αφού αν για παράδειγμα σε μια web εφαρμογή όταν δεν υπάρχει επισκεψιμότητα δεν χρειάζεται να τρέχουν περισσότεροι servers ή μεγαλύτεροι servers. Όταν η επισκεψιμότητα αυξηθεί και το σύστημα χρειαστεί περισσότερους πόρους τότε αυτόματα για εκείνη την στιγμή το σύστημα θα μπορεί να κλιμακωθεί ανάλογα.

4.2 Οριζόντια και Κάθετη κλιμάκωση συστημάτων

Κλιμάκωση είναι η διαδικασία κατά την οποία ένα σύστημα αλλάζει τους διαθέσιμους πόρους της. Η διαφορά μεταξύ οριζόντιας και κάθετης κλιμάκωσης είναι ότι στην πρώτη αλλάζει ο αριθμός του συνόλου των όμοιων οντοτήτων ενώ στην δεύτερη αλλάζουν οι τιμές των χαρακτηριστικών των υπάρχοντων οντοτήτων. Δηλαδή αν είχαμε έναν εξυπηρετητή με διαθέσιμη μνήμη 2GB κατά την οριζόντια κλιμάκωση θα προστίθεται ακόμα ένας εξυπηρετητής με τα ίδια χαρακτηριστικά ενώ κατά κάθετη κλιμάκωση η μνήμη του εξυπηρετητή θα άλλαζε από 2GB σε 8GB.

4.3 Εικονικοποίηση (Virtualization)

Στην επιστήμη της πληροφορικής, η εικονικοποίηση (virtualization) είναι ένας ευρύς όρος των υπολογιστικών συστημάτων που αναφέρεται σε έναν μηχανισμό αφαίρεσης, στοχευμένο στην απόκρυψη λεπτομερειών της υλοποίησης και της κατάστασης ορισμένων υπολογιστικών πόρων από πελάτες των πόρων αυτών (π.χ. εφαρμογές, άλλα συστήματα, χρήστες κλπ). [14]

Η εν λόγω αφαίρεση μπορεί είτε να αναγκάζει έναν πόρο να συμπεριφέρεται ως πλειάδα πόρων (π.χ. μία συσκευή αποθήκευσης σε διακομιστή τοπικού δικτύου), είτε πολλαπλούς πόρους να συμπεριφέρονται ως ένας (π.χ. συσκευές αποθήκευσης σε καταναμημένα συστήματα).

Η εικονικοποίηση δημιουργεί μία εξωτερική διασύνδεση η οποία αποκρύπτει την υποκείμενη υλοποίηση (π.χ. πολυπλέκοντας την πρόσβαση από διαφορετικούς χρήστες). Αυτή η προσέγγιση στην εικονικοποίηση αναφέρεται ως εικονικοποίηση πόρων. Μία άλλη προσέγγιση, ίδιας όμως νοοτροπίας, είναι η εικονικοποίηση πλατφόρμας, όπου η αφαίρεση που επιτελείται προσομοιώνει ολόκληρους υπολογιστές. Το αντίθετο της εικονικοποίησης είναι η διαφάνεια: ένας εικονικός πόρος είναι ορατός, αντιληπτός, αλλά στην πραγματικότητα ανύπαρκτος, ενώ ένας διαφανής πόρος είναι υπαρκτός αλλά αόρατος.

4.4 Εικονικοποίηση Πλατφόρμας

Ιδιαίτερο ενδιαφέρον παρουσιάζει η εικονικοποίηση πλατφόρμας, όπου ένα λογισμικό ελέγχου («επόπτης» ή hypervisor) εκτελούμενο σε πραγματικό υλικό προσομοιώνει ένα υπολογιστικό περιβάλλον, μία «εικονική μηχανή», επάνω από το οποίο μπορεί να τρέξει κάποιο φιλοξενούμενο λογισμικό (συνήθως ένας πλήρης πυρήνας), απομονωμένο από το υπόλοιπο σύστημα. [14]

Η θεμελιώδης λογική πίσω από την εικονικοποίηση πλατφόρμας είναι η αρχή πως οποιαδήποτε λειτουργία μπορεί να εκτελεστεί είτε από λογισμικό είτε από εξειδικευμένο υλικό· οι μόνες διαφορές αφορούν την ευελιξία και την απόδοση.

Είναι δυνατόν να προσομοιώνονται ταυτόχρονα πολλαπλές εικονικές μηχανές, εντελώς απομονωμένες μεταξύ τους, από το ίδιο λογισμικό ελέγχου.

Η εικονικοποίηση πλατφόρμας εμφανίστηκε αρχικά τη δεκαετία του 1960, πριν από την επέλαση των μικροϋπολογιστών, σε μεγάλα, συγκεντρωτικά συστήματα

(mainframes), αλλά μετά το 2000 και την αλματώδη αύξηση των επιδόσεων του υλικού των PC έχει γίνει πλέον κοινή πρακτική.

4.5 Περιγραφή του Docker

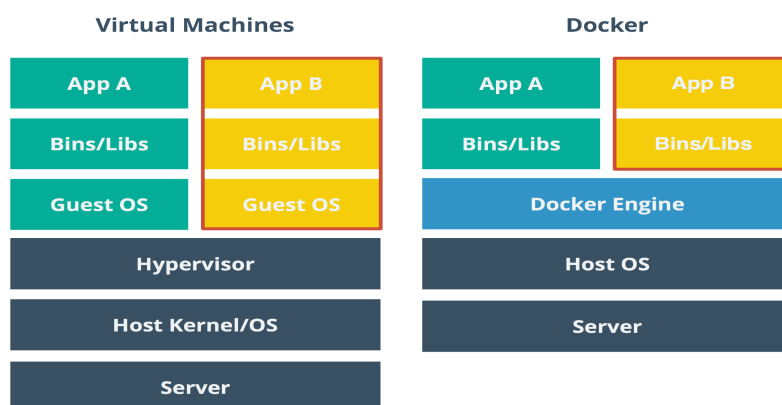
Το Docker (Ντόκερ) είναι μια πλατφόρμα λογισμικού ανοιχτού κώδικα που υλοποιεί Εικονικοποίηση (Virtualization) σε επίπεδο Λειτουργικού Συστήματος. Ουσιαστικά το Docker προσφέρει αυτοματοποιημένες διαδικασίες για την ανάπτυξη εφαρμογών σε απομονωμένες Περιοχές Χρήστη (User Spaces) που ονομάζονται Software Containers. [4]

Το λογισμικό χρησιμοποιεί τεχνολογίες του πυρήνα του Linux όπως τα cgroups και οι χώροι ονομάτων πυρήνα (kernel namespaces), για να επιτρέψει σε ανεξάρτητα software containers να εκτελούνται στο ίδιο λειτουργικό σύστημα.

Έτσι αποφεύγεται η χρήση επιπλέον υπολογιστικών πόρων που θα απαιτούσε μια εικονική μηχανή (virtual machine).

Το Docker είναι μηχανισμός ανοιχτού κώδικα, ο κύριος στόχος του οποίου είναι να αυτοματοποιήσει την διανομή εφαρμογών μέσα σε δοχεία λογισμικού και την αυτοματοποίηση του Virtualization των μικρο-υπηρεσιών του λειτουργικού συστήματος στο Linux.

Τα docker containers δημιουργούνται από στιγμιότυπα (images) docker. Μπορείτε να φανταστείτε ένα docker container ως τη ζωντανή κατάσταση μιας διαδικτυακής εφαρμογής που τρέχει από ένα αρχείο ISO. Αλλά αυτή τη φορά το ISO, το οποίο στο παράδειγμά μας είναι το ισοδύναμο του στιγμιότυπου του docker image, περιέχει μόνο την εφαρμογή και τις εξαρτήσεις της. Ένα μεγάλο χαρακτηριστικό του docker είναι το αρχείο docker (docker file). Ένα **αρχείο docker** είναι η **συνταγή** που περιέχει όλα τα απαραίτητα βήματα που απαιτούνται για τη δημιουργία μιας εικόνας docker.



Εικόνα 13 Σύγκριση Docker με εικονικές μηχανές

4.6 Εγκατάσταση Docker

Με τις παρακάτω εντολές γίνεται η εγκατάσταση του Docker στο σύστημά μας καθώς επίσης αφού ξεκινήσουμε την υπηρεσία δίνουμε και τα κατάλληλα δικαιώματα στον χρήστη μας για να μπορεί να χρησιμοποιεί το Docker. [4]

```
sudo apt-get update

sudo apt-get install docker-ce

service docker start

usermod -a -G docker myuser
```

Έχοντας εξηγήσει τι είναι τα docker files παραπάνω, τα Docker Containers και τα Docker Images, ένας τρόπος για να τα οργανώσουμε όλα μαζί για να τα χρησιμοποιήσουμε σε ένα μια εφαρμογή είναι το λογισμικό docker-compose.

Το docker-compose διαβάσει από ένα αρχείο παραμετροποίησης που ονομάζεται **docker-compose.yml** και περιγράφει ποια docker containers θα τρέξουν σαν υπηρεσίες (services) αφού δημιουργηθούν από τα αντίστοιχα docker images όπως αυτά περιγράφονται στο κάθε Docker file.

Παρακάτω φαίνεται το docker-compose.yml αρχείο της εφαρμογής.

```
version: '2'
services:
  # Nginx
  webserver:
    image: nginx:1.11
    depends_on:
      - php
      - db
    links:
      - php
    volumes:
      - ./app:/var/www/html/app
    volumes:
      - ./docker/nginx/app.conf:/etc/nginx/conf.d/default.conf:ro
    ports:
      - 8080:80
  # PHP -FPM
  php:
    build: ./docker/php/
    environment:
      TIMEZONE: Europe/Athens
```

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

```
volumes:
  - ./docker/php/php.ini:/usr/local/etc/php/php.ini:ro
  - ./app:/var/www/html/app
working_dir: /var/www/html/app

# MySQL
db:
  build: ./docker/mysql
  volumes:
    - ./mysql_data:/var/lib/mysql
  restart: unless-stopped
  ports:
    - "4983:3306"
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: api
    MYSQL_USER: root
    MYSQL_PASSWORD: root

# Composer
composer:
  image: composer:1.4
  volumes_from:
    - php
  working_dir: /var/www/html/app
```

Για να δούμε όλα τα docker containers που τρέχουν για την εφαρμογή αρκεί να εκτελέσουμε την παρακάτω εντολή:

```
docker ps
```

ενώ για να δούμε όλα τα Docker Images της εφαρμογής θα πρέπει να εκτελέσουμε :

```
docker-compose images
```

4.7 Περιγραφή τηςσχεσιακής βάσης δεδομένων MySQL

Για την σχεσιακή βάση δεδομένων χρησιμοποιήθηκε η **MySQL** με έκδοση **5.7**. Είναι ανοιχτού κώδικα λογισμικό υπό την άδεια GNU General Public License, γραμμένη σε C και C++ και ξεκίνησε το 1995 από τους Michael Widenius και David Axmark.

Στην παρακάτω εικόνα φαίνεται το σχήμα της βάσης (Schema) και ακολουθούν μερικά βασικά queries για να κατανοήσουμε καλύτερα την δομή της βάσης δεδομένων της εφαρμογής.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.



Εικόνα 13 Σχήμα της βάσης δεδομένων της εφαρμογής

Για να έχουμε σαν αποτέλεσμα την λίστα όλων των χρηστών της βάσης μαζί με τις επιπλέον πληροφορίες για αυτούς.

```
SELECT *  
FROM fos_user  
INNER JOIN user_info ON fos_user.id = user_info.user_id
```

Για να έχουμε σαν αποτέλεσμα την λίστα όλων των λογαριασμών ενός χρήστη.

```
SELECT *  
FROM account  
WHERE user_id = 1
```

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Για να έχουμε σαν αποτέλεσμα την λίστα όλων των καρτών που έχει ένας χρήστης.

```
SELECT *  
FROM card  
WHERE card.user_id =1;
```

Για να έχουμε σαν αποτέλεσμα την λίστα όλων των κινήσεων ενός τραπεζικού λογαριασμού.

```
SELECT *  
FROM TRANSACTION  
WHERE transaction.account_id = 1
```

Για να έχουμε σαν αποτέλεσμα την λίστα όλων των κινήσεων ενός χρήστη ανεξάρτητα τραπεζικού λογαριασμού.

```
SELECT *  
FROM TRANSACTION  
INNER JOIN account ON account.account_id = transaction.account_id  
AND account.user_id = 1
```

Dockerfile για το MySql Docker Image (/docker/MySql/Dockerfile)

```
FROM mysql:5.7  
  
RUN usermod -u 1000 mysql  
RUN mkdir -p /var/run/mysqld  
RUN chmod -R 777 /var/run/mysqld
```

4.8 Περιγραφή της γλώσσας προγραμματισμού PHP

Η PHP (PHP: Hypertext Preprocessor) είναι μια γλώσσα προγραμματισμού για τη δημιουργία σελίδων web με δυναμικό περιεχόμενο αλλά και γενικού σκοπού και ξεκίνησε το 1994 από τον Rasmus Lerdorf.

Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού (στην περίπτωση μας Nginx), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που είτε θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών (Browser) είτε σε μορφή κώδικα HTML είτε σε αποτελέσματα με σκοπό την επικοινωνία συστημάτων (στην περίπτωση μας Json) , ή θα επεξεργασθεί τις εισόδους δίχως να προβάλλει την έξοδο στο χρήστη, αλλά θα τις μεταβιβάσει σε κάποιο άλλο PHP script.

Η PHP αρχικά χρησιμοποιεί το **CGI (Common Gateway Interface)** για να επικοινωνήσει με τον web server. Το **CGI** είναι μια διεπαφή που λέει στον webserver πώς να μεταφέρει δεδομένα από και προς μια εφαρμογή. [8]

Ο FPM (FastCGI Process Manager) αντικαθιστά το μοντέλο υλοποίησης της PHP με χρήση CGI, και περιέχει περισσότερες λειτουργίες καθώς και μπορεί να χρησιμοποιηθεί για εφαρμογές με βαρύ φορτίο σε αιτήσεις (heavy load).

Η έκδοση της PHP που χρησιμοποιήθηκε για την εφαρμογή είναι η **7.1.9** και παρακάτω φαίνεται το αρχείο Dockerfile για την δημιουργία του Docker Image της PHP και η εγκατάσταση των απαραίτητων βιβλιοθηκών συστήματος που απαιτούνται για να λειτουργήσει το Symfony framework.

Dockerfile για το PHP_FPM Docker Image (/docker/php/Dockerfile)

```
FROM php:7.1.9-fpm

# Install recommended extensions for Symfony
RUN apt-get update && apt-get install -y \
    libc-dev \
    libmcrypt-dev \
    libpq-dev \
    libbz2-dev \
    php-pear \
    && docker-php-ext-install \
    intl \
    opcache \
    && docker-php-ext-install pdo_mysql \
    && docker-php-ext-enable \
    intl \
    opcache

# Permission fix
RUN usermod -u 1000 www-data
RUN sed -ri 's/^www-data:x:82:82:/www-data:x:1000:50:/' /etc/passwd

ENV PHP_EXTRA_CONFIGURE_ARGS --enable-fpm --with-fpm-user=www-data --with-fpm-group=www-data
```

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

```
#RUN chown -R www-data:www-data /var/www/app/var/cache
#RUN chown -R www-data:www-data /var/www/app/var/Logs
RUN pwd
RUN whoami
```

Αρχείο php.iniγια την παραπμετροποίηση της PHP. (/docker/php/php.ini)

```
date.timezone = ${TIMEZONE}
short_open_tag = Off
log_errors = On
error_reporting = E_ALL
display_errors = Off
error_log = /proc/self/fd/2
memory_limit = 256M

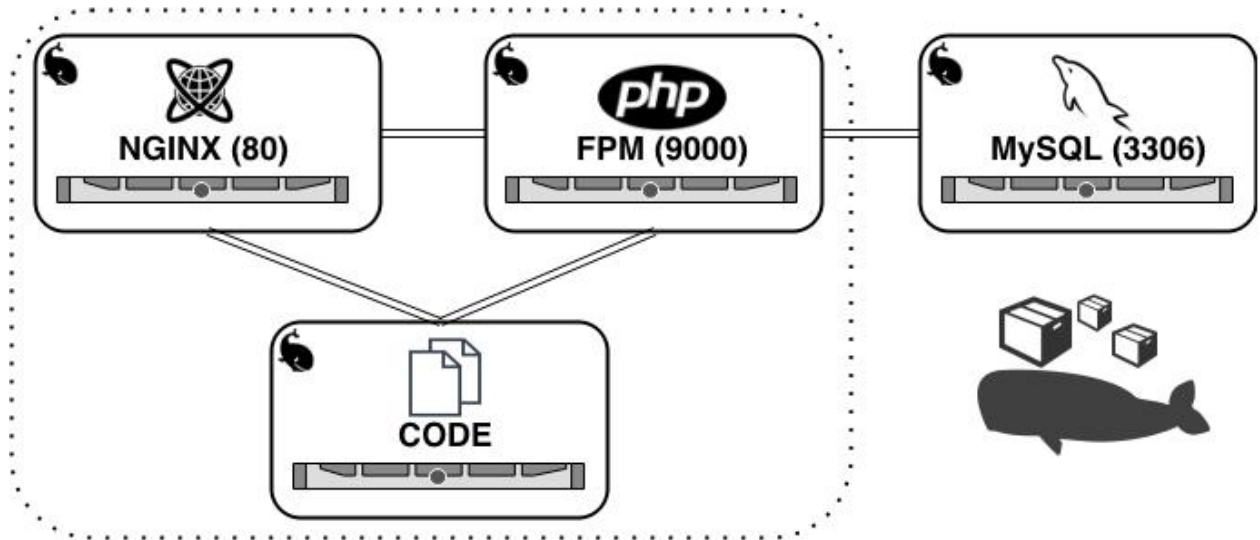
; Optimizations for Symfony, as documented on
http://symfony.com/doc/current/performance.html
opcache.max_accelerated_files = 20000
realpath_cache_size = 4096K
realpath_cache_ttl = 600
```

4.9 Περιγραφή του εξυπηρετητή ιστού Nginx

Ο εξυπηρετητής ιστού (web server) της εφαρμογής και υπεύθυνος για τα Requests είναι ο **Nginx web server** με έκδοση **1.11**. Ο Nginx αποτελεί έναν δυνατό και επεκτάσιμο γενικού σκοπού εξυπηρετητή ιστού αλλά με τον οποίο επίσης να μπορεί να χρησιμοποιηθεί και ως εξ ισορροπιστή φορτίου (load balancer), δικτυακό διαμεσολαβητή (proxy), μεσολαβητή ηλεκτρονικού ταχυδρομείου (mail proxy) και αλλά και ενδιάμεσης προσωρινής μνήμης (web cache). Είναι ανοικτού κώδικα λογισμικό υπό την άδεια BSD-like και ξεκίνησε το 2004 από τον Igor Sysoev. [5]

Παρακάτω φαίνεται η παραμετροποίηση του Nginx για να μπορέσει να εξυπηρετήσει την εφαρμογή σύμφωνα με τις απαιτήσεις του Symfony Framework.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.



Εικόνα 14 Περιγραφή συνδιασμού χρήσης MySql Nginx PHP/FPM

Οι βασικές ρυθμίσεις είναι ότι εξυπηρετεί τις αιτήσεις στην πόρτα (port) 80 με αρχική τοποθεσία των αρχείων που εξυπηρετεί (root path) στην τοποθεσία **/var/www/html/app/web** και για την χρήση της php γίνεται η επικοινωνία με τον fpm στην δικτυακή πόρτα 9000.

```
upstream php-upstream {
    server php:9000;
}

server {
    root /var/www/html/app/web;
    listen 80;
    server_tokens off;

    location / {
        try_files $uri @rewriteapp;
    }

    location @rewriteapp {
        rewrite ^(.*)$ /app.php/$1 last;
    }

    location ~ ^/(app|app_dev|app_test|config)\.php(/|$) {
        fastcgi_pass php-upstream;
        fastcgi_split_path_info ^(.+\.\php)(/.*)$;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param HTTPS off;
    }
}
```

ΚΕΦΑΛΑΙΟ 5

Amazon Web Services

5.1 Περιγραφή του διακομιστή υπηρεσιών cloud AWS

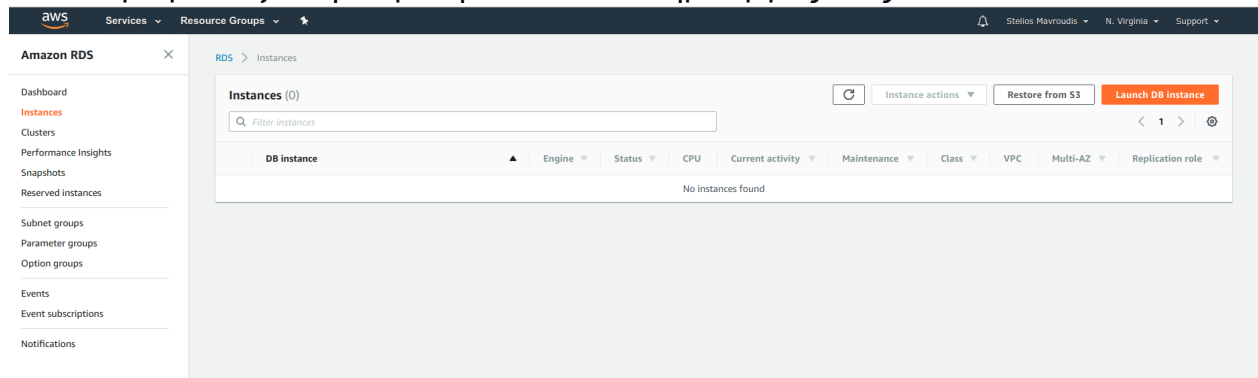
Η Amazon αποτελεί έναν από τους μεγαλύτερους διακομιστές υπηρεσιών Cloud παγκοσμίως αναπτύσσοντας συνεχώς και υλοποιώντας τρόπους για να μπορεί να γίνει ασφαλέστερη , γρηγορότερη και ακόμα πιο σταθερή η ανάπτυξη και η διαθεσιμότητα μιας εφαρμογής στους χρήστες. Παρακάτω θα αναλύσουμε μερικές από τις τεχνολογίες που χρησιμοποιήθηκαν για την διπλωματική εργασία. [3]

5.2 Amazon Relational Database Service (RDS)

Είναι η υπηρεσία κατά την οποία μπορούμε εύκολα να δημιουργήσουμε και να διαχειριστούμε σχεσιακές βάσεις δεδομένων στο Cloud. Υποστηρίζει τις ακώλουθες βάσεις δεδομένων Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, και Microsoft SQL Server. Μπορούμε εύκολα να δημιουργήσουμε εφεδρική βάση δεδομένων σαν κλώνο της υπάρχουσας (DB Replication) , να κρατήσουμε στιγμιότυπα (snapshots) με την εκάστοτε κατάσταση της βάσης, να παρακολουθούμε την απόδοση και το φορτίο ανά πάσα στιγμή αλλά και να προσαρμόσουμε τα χαρακτηριστικά της βάσης μας ανάλογα με τις ανάγκες μας (μεγαλύτερη μνήμη RAM, μεγαλύτερη χωρητικότητα δίσκου κ.λπ). [9]

Παρακάτω φαίνεται βήμα βήμα πως δημιουργήθηκε ένα RDS με MySql για την διπλωματική μέσω της διαδικτυακής κονσόλας (web console) της Amazon.

1.Επιλέγουμε να ξεκινήσουμε την διαδικασία δημιουργίας ενός RDS.



Εικόνα 15 Δημιουργία RDS - βήμα 1

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

2. Στην συνέχεια επιλέγουμε την MySQL για σχεσιακή βάση δεδομένων

The screenshot displays the 'Specify DB details' page in the AWS RDS console. The breadcrumb navigation at the top reads 'RDS > Instances > Launch DB instance'. On the left, a sidebar shows three steps: 'Step 1 Select engine', 'Step 2 Specify DB details' (which is the active step), and 'Step 3 Configure advanced settings'. The main content area is titled 'Specify DB details' and includes the following sections:

- Instance specifications**: A note to estimate monthly costs using the AWS Simple Monthly Calculator.
- DB engine**: Set to 'MySQL Community Edition'.
- License model**: Set to 'general-public-license'.
- DB engine version**: Set to 'mysql 5.7.21'.
- Known Issues/Limitations**: A warning box advising to review compatibility issues with specific database versions.
- Free tier**: A warning box explaining the Amazon RDS Free Tier (single db.t2.micro instance, up to 20 GiB storage) and a checked checkbox for 'Only enable options eligible for RDS Free Usage Tier'.
- DB instance class**: Set to 'db.t2.micro — 1 vCPU, 1 GiB RAM'.
- Multi-AZ deployment**: A radio button option for 'Create replica in different zone' is selected, with a note that it provides data redundancy and reduces latency during backups.

Εικόνα 16 Δημιουργία RDS βήμα 2

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

3. Ορίζουμε το αναγνωριστικό όνομα του RDS καθώς και τον χρήστη που θα είναι ο διαχειριστής της βάσης. Οι ρυθμίσεις που ακολουθούν αφορούν την δημιουργία του RDS στο ήδη υπάρχων εικονικό δίκτυο (VPC).

The screenshot shows the 'Settings' section of the AWS RDS console. At the top, there is a field for 'Allocated storage' set to '20 GiB'. Below this, the 'DB instance identifier' is set to 'MyRds'. The 'Master username' is 'dbadmin', and the 'Master password' and 'Confirm password' fields are filled with masked characters. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons.

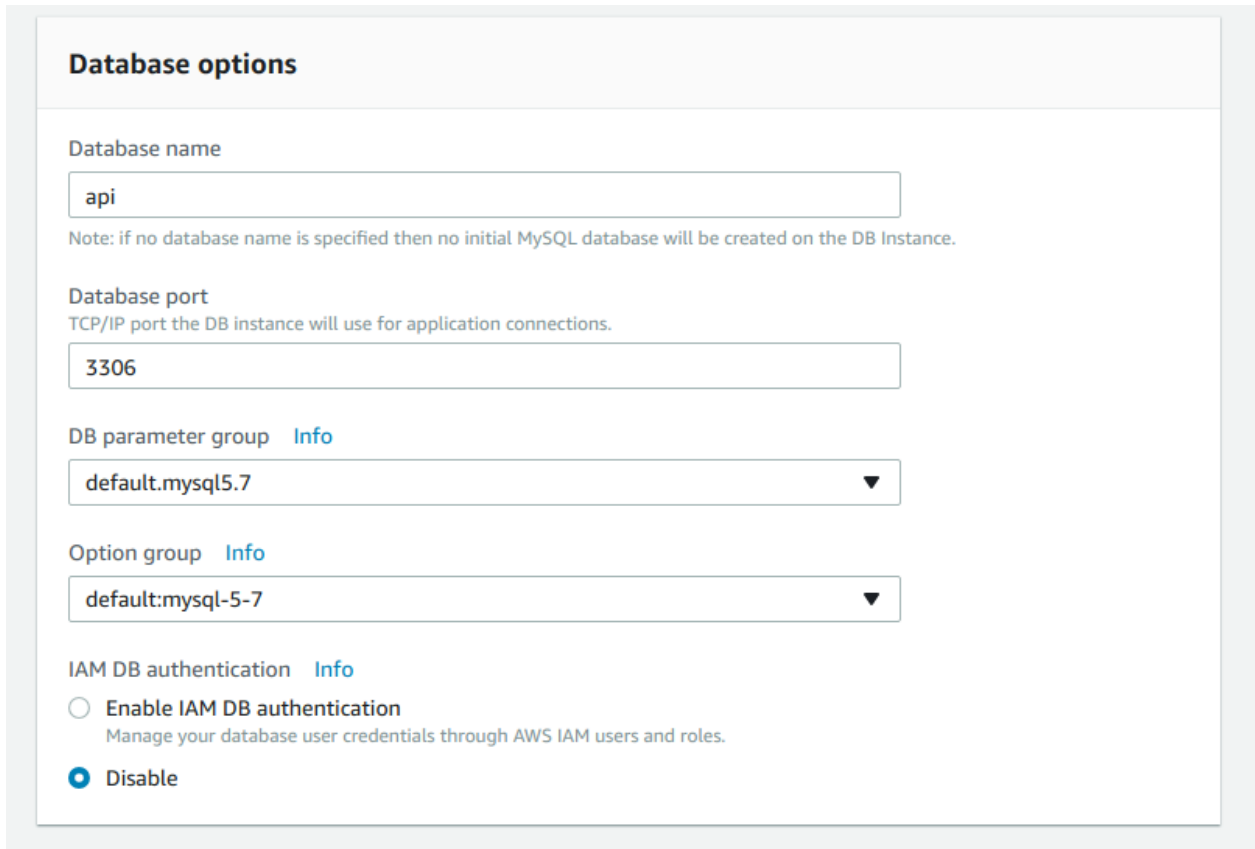
Εικόνα 17 Δημιουργία RDS βήμα 3

The screenshot shows the 'Configure advanced settings' section of the AWS RDS console. The 'Network & Security' section is expanded, showing 'Virtual Private Cloud (VPC)' set to 'Default VPC (vpc-ee96b597)', 'Subnet group' set to 'default', 'Public accessibility' set to 'Yes', 'Availability zone' set to 'No preference', and 'VPC security groups' set to 'Create new VPC security group'. The left sidebar shows the progress through the steps: 'Step 1 Select engine', 'Step 2 Specify DB details', and 'Step 3 Configure advanced settings'.

Εικόνα 18 Δημιουργία RDS βήμα 4

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

3. Ορίζουμε το όνομα της βάση δεδομένων που θα έχει το RDS. Ένα RDS μπορεί να φιλοξενήσει πολλές διαφορετικές βάσεις δεδομένων.



Database options

Database name
api
Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Database port
TCP/IP port the DB instance will use for application connections.
3306

DB parameter group [Info](#)
default.mysql5.7

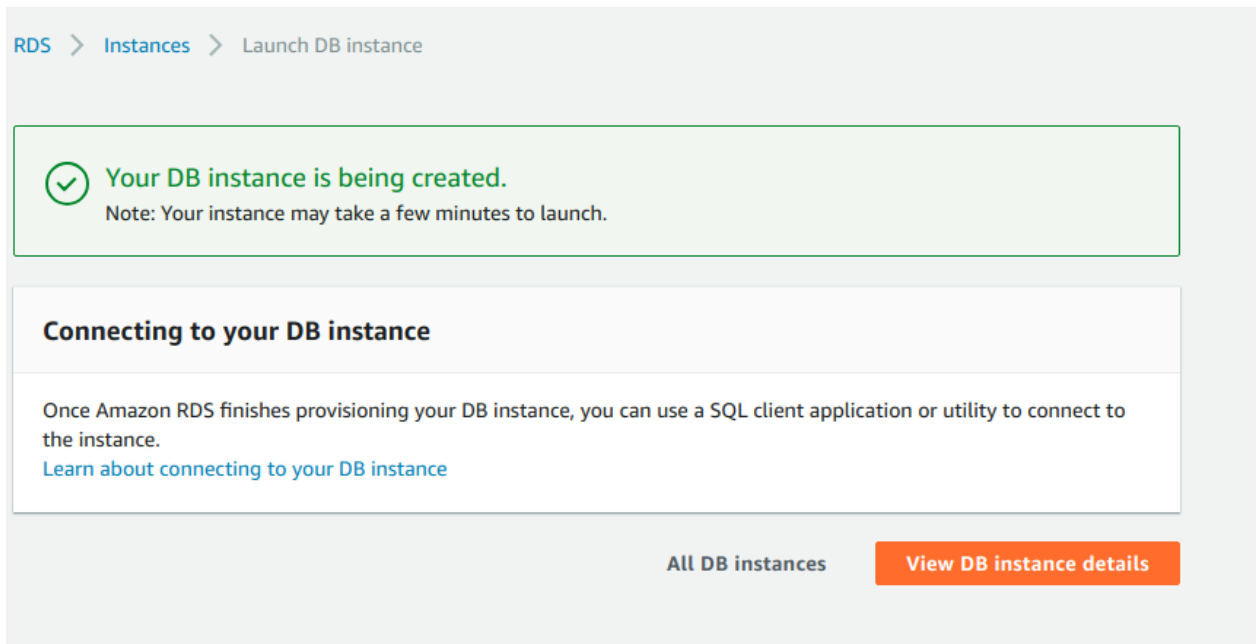
Option group [Info](#)
default:mysql-5-7

IAM DB authentication [Info](#)
 Enable IAM DB authentication
Manage your database user credentials through AWS IAM users and roles.
 Disable

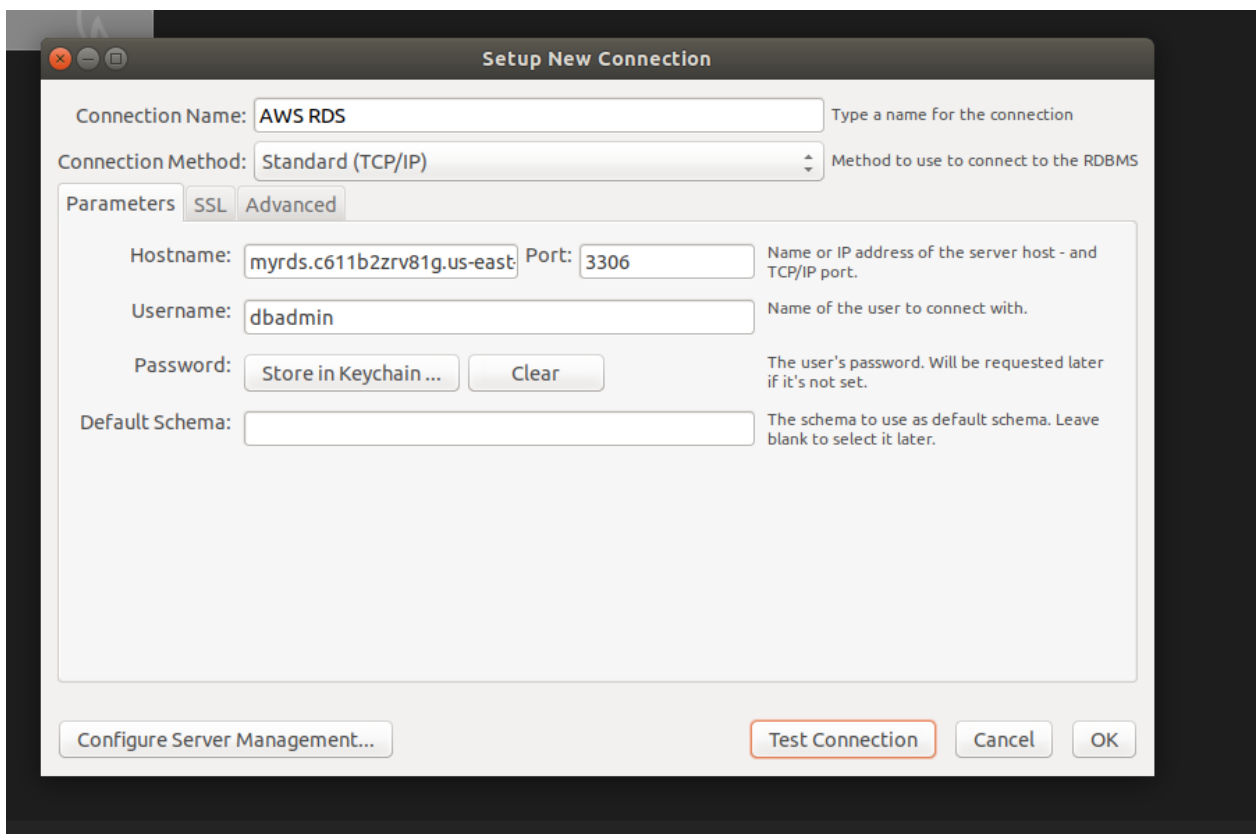
Εικόνα 19 Δημιουργία RDS βήμα 5

4. Έχοντας κάνει όλα τα βήματα σωστά δοκιμάζουμε να συνδεθούμε απομακρυσμένα στο RDS χρησιμοποιώντας την διαθέσιμη προς το κοινό (public) διεύθυνση του RDS.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.



Εικόνα 20 Δημιουργία RDS βήμα 6



Εικόνα 21 Έλεγχος απομακρυσμένης σύνδεσης με το RDS

5.3 Elastic Compute Cloud (Amazon EC2)

Ένα από τα πιο χαρακτηριστικά ενός διακομιστή Cloud υπηρεσιών είναι η δυνατότητα δημιουργίας εικονικού υπολογιστή έχοντας όλα τα κοινά χαρακτηριστικά που έχει ένας κοινός υπολογιστής (Μνήμη RAM, μονάδα επεξεργασίας/ CPU κ.λπ)

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

μόνο που αυτά είναι εικονικά και καταλάβουν αντίστοιχους πόρους από το φυσικό υλικό ενός φυσικού υπολογιστή. [10]

Το Amazon EC2 παρέχει στους προγραμματιστές τα εργαλεία για την κατασκευή ανεπαρκών ελαστικών εφαρμογών και την αποκλεισμό τους από κοινά σενάρια αποτυχίας.

5.4 Amazon Simple Storage Service (Amazon S3)

Αποτελεί έναν από τον καλύτερο τρόπο αποθήκευσης και διαμοιρασμού αρχείων στο Cloud. Μπορούμε να χρησιμοποιήσουμε πολιτικές για να βελτιστοποιήσουμε το κόστος αποθήκευσης, συνδυάζοντας αυτόματα τις διαφορετικές κατηγορίες αποθήκευσης. Το AWS καθιστά την αποθήκευση ευκολότερη στη χρήση για να κάνει ανάλυση, να αποκτήσει γνώσεις και να πάρει καλύτερες αποφάσεις γρηγορότερα. [11]

Σε παρακάτω ενότητα θα δούμε πώς χρησιμοποιήθηκαν οι υπηρεσίες EC2 και S3 για την συγκεκριμένη διπλωματική εργασία.

ΚΕΦΑΛΑΙΟ 6

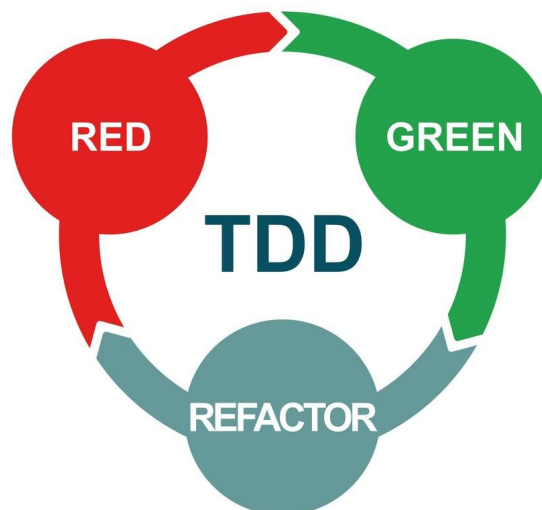
Βοηθητικές μεθοδολογίες ανάπτυξης λογισμικού

6.1 Περιγραφή της μεθοδολογίας ανάπτυξης λογισμικού Test Driven Development

Είναι η μεθοδολογία ανάπτυξης λογισμικού ξεκινώντας πρώτα με την υλοποίηση των test σεναρίων σε κώδικα με σκοπό το αποτέλεσμα που θέλουμε να φτάσουμε.

Τα τεστ σενάρια κώδικα με διάφορα εργαλεία ανάλογα την γλώσσα προγραμματισμού που χρησιμοποιούμε “σπάνε” δηλαδή δεν ικανοποιούνται από τα αποτελέσματα που έχουμε ορίσει ότι περιμένουμε να έχουμε. Έτσι θα πρέπει ο προγραμματιστής να φτιάξει τον κώδικα ώστε το συγκεκριμένο τεστ σενάριο που σπάει να γίνει “πράσινο”. Η τεχνική αυτή χαρακτηρίζεται και ως Red-Green-Refactor δηλαδή από μια κόκκινη κατάσταση όπου τα τεστ σενάρια κώδικα βγάζουν σφάλματα να διαμορφώσουμε έτσι τον κώδικα ώστε να γίνουν πράσινα και να έχουν σφάλματα.

Η διαδικασία του refactor μπορεί να υλοποιηθεί όσο ο κώδικάς μας μπορεί να βελτιστοποιηθεί. Τα τεστ σενάρια κώδικα εκτός από την ανάπτυξη του κώδικα είναι εξαιρετικά χρήσιμα και για τον έλεγχο της σωστής λειτουργίας της εφαρμογής. Έτσι όταν ένας προγραμματιστής γράψει ένα κομμάτι κώδικα που επηρεάζει λάθος κάποιο άλλο το αντίστοιχο τεστ σενάριο θα σπάσει και πρέπει να διορθωθεί η λειτουργικότητα του κώδικα. Τέλος τα τεστ σενάρια κώδικα μπορούν να προλάβουν έναν λανθασμένο κώδικα να ανέβει σε μια εφαρμογή που είναι διαθέσιμη στο κοινό.



Εικόνα 22 Περιγραφή μεθοδολογίας TDD

6.2 Περιγραφή της μεθοδολογίας Continuous Integration(CI).

Στην ανάπτυξη λογισμικού ο όρος Continuous Integration αποτελεί την τεχνική κατά την οποία ταυτόχρονα πολλοί διαφορετικοί μηχανικοί πληροφορικής και προγραμματιστές να δουλεύουν τα δικά τους κομμάτια κώδικα και να τα ανανεώνουν στο υπάρχον διαθέσιμο λογισμικό προς το κοινό (production environment) πολλές φορές μέσα στην ίδια μέρα με την τελευταία διαδικασία να μην είναι διακριτή προς τους χρήστες σταματώντας για παράδειγμα την λειτουργία της εφαρμογής την ώρα που πραγματοποιείται η ανανέωση της.

Με την παραπάνω διαδικασία και την βοήθεια αρκετών εργαλείων πετυχαίνουμε την γρηγορότερη παράδοση μεμονωμένων κομματιών της εφαρμογής στον πελάτη αλλά και την ταχύτερη διόρθωση σφαλμάτων όταν υπάρξουν.

Όλοι οι προγραμματιστές ακολουθούν τον ίδιο τρόπο για να παραδώσουν την δουλειάς ξεχωριστά ή ταυτόχρονα χωρίς το τελευταίο να δημιουργεί προβλήματα

Στην συνέχεια φαίνεται βήμα βήμα πως χρησιμοποιήθηκαν διάφορες υπηρεσίες της Amazon (Amazon Web Services) για να υλοποιηθεί η συγκεκριμένη τεχνική για την εφαρμογή της διπλωματικής

Με την βοήθεια της υπηρεσίας CodeDeploy της AWS (Amzon Web Services) θα αυτοματοποιήσουμε την διαδικασία του deploy της εφαρμογής στα εικονικά μηχανήματα που παίζουν τον ρόλο των διακομιστών.

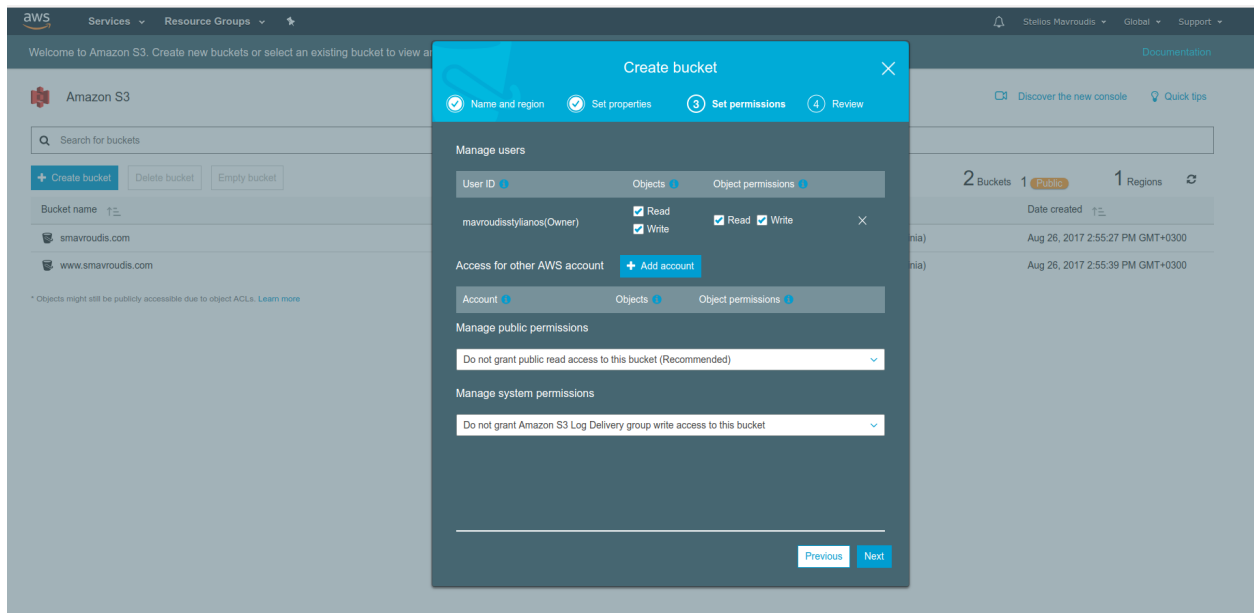
Πρώτα θα πρέπει να έχουμε δημιουργήσει τις αντίστοιχες οντότητες μέσω της διαδικτυακής κονσόλας της AWS οι οποίες είναι απαραίτητες για την παραμετροποίηση του της υπηρεσίας CodeDeploy.

6.3 Δημιουργία s3 bucket

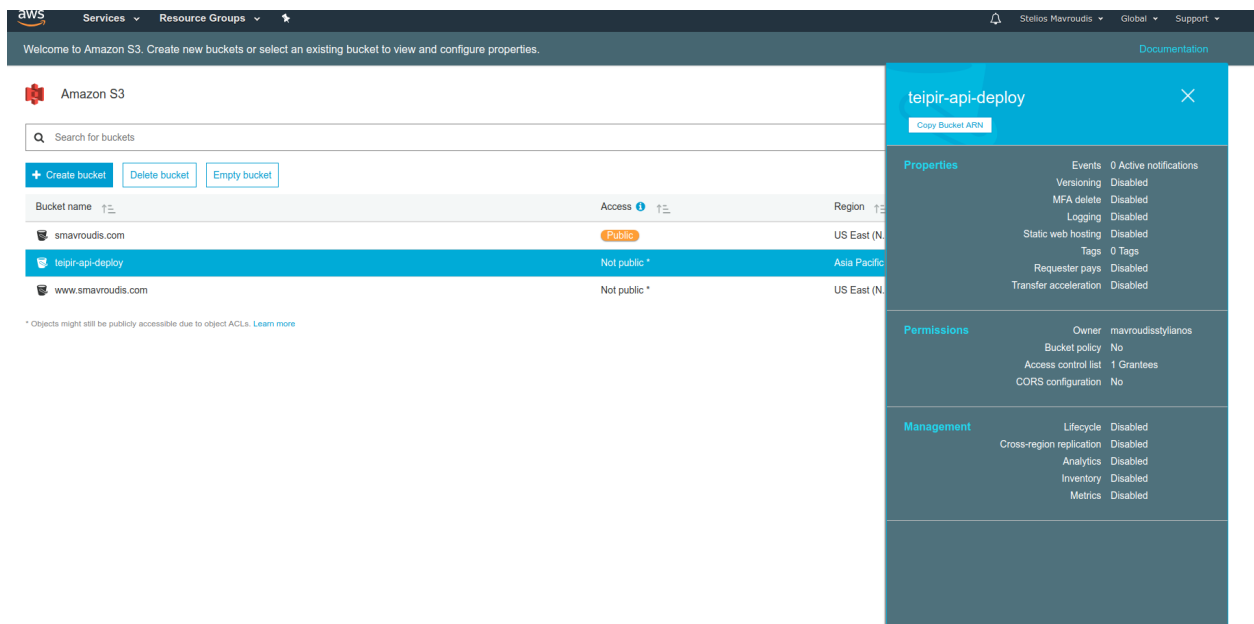
Το CodeDeploy θα δούμε και παρακάτω ότι φτιάχνει ένα συμπιεσμένο αρχείο (zip file) με περιεχόμενο όλο τον κώδικα της εφαρμογής και αφού το πακετάρει, ανεβάζει αυτό το αρχείο στην υπηρεσία αποθήκευσης στο Cloud της Amazon (S3).

Στο συμπιεσμένο αυτό αρχείο προστίθεται και ένα μοναδικό αναγνωριστικό και ορίζει την εκάστοτε έκδοση της εφαρμογής μας.

Για την εφαρμογή της διπλωματικής το s3 bucket (κουβάς αρχείων) που χρησιμοποιήθηκε είναι το teipir-api-deploy.



Εικόνα 23 Δημιουργία S3 βήμα 1



Εικόνα 24 Λίστα s3

6.4 Δημιουργία χρήστη συστήματος

Η Amazon έχει αναπτύξει ισχυρό σύστημα για την δημιουργία και διαχείριση χρηστών για την διαχείριση των υπηρεσιών της. Μπορούμε εκτός από χρήστες που ανήκουν σε διαχειριστές συστήματος να φτιάξουμε λογαριασμούς χρηστών για να χρησιμοποιεί το ίδιο το σύστημα (χρήστης συστήματος).

Οι λογαριασμοί χρηστών συστήματος (όπως και οι απλοί λογαριασμοί όμως) συνοδεύονται απαραίτητα από ένα τουλάχιστον ζευγάρι αναγνωριστικών κλειδιών (Access key ID και Secret access key).

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Στους λογαριασμούς χρηστών ανάλογα με τις λειτουργίες που πρέπει να εκτελέσουν χρησιμοποιώντας τις διάφορες υπηρεσίες δίνονται και τα αντίστοιχα δικαιώματα ή ρόλοι.

Για την υπηρεσία CodeDeploy θα δημιουργήσουμε έναν χρήστη συστήματος με όνομα **tei_pir_deployer** και θα του δώσουμε τα αντίστοιχα δικαιώματα όπως για παράδειγμα να μπορεί να αποθηκεύει στο s3 τις καινούριες εκδόσεις της εφαρμογής ώστε να μεταβούν έπειτα στα μηχανήματα.

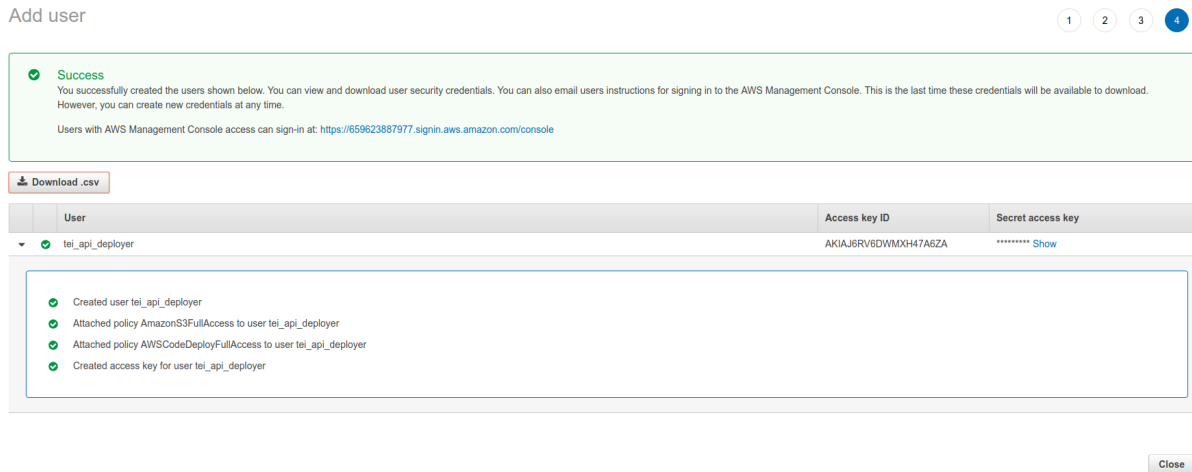
The screenshot shows the 'Add user' page in the AWS IAM console. The user name is 'tei_api_deployer'. The 'Access type' is set to 'Programmatic access', which enables an access key ID and secret access key. The 'AWS Management Console access' option is not selected. The page includes a 'Cancel' button and a 'Next: Permissions' button.

Εικόνα 25 Δημιουργία χρήστη συστήματος βήμα 1

The screenshot shows the 'Add user' page in the AWS IAM console, step 3: Review. The user name is 'tei_api_deployer' and the access type is 'Programmatic access - with an access key'. The 'Permissions summary' section shows two managed policies attached to the user: 'AmazonS3FullAccess' and 'AWSCodeDeployFullAccess'. The page includes 'Cancel', 'Previous', and 'Create user' buttons.

Type	Name
Managed policy	AmazonS3FullAccess
Managed policy	AWSCodeDeployFullAccess

Εικόνα 26 Δημιουργία χρήστη συστήματος βήμα 2



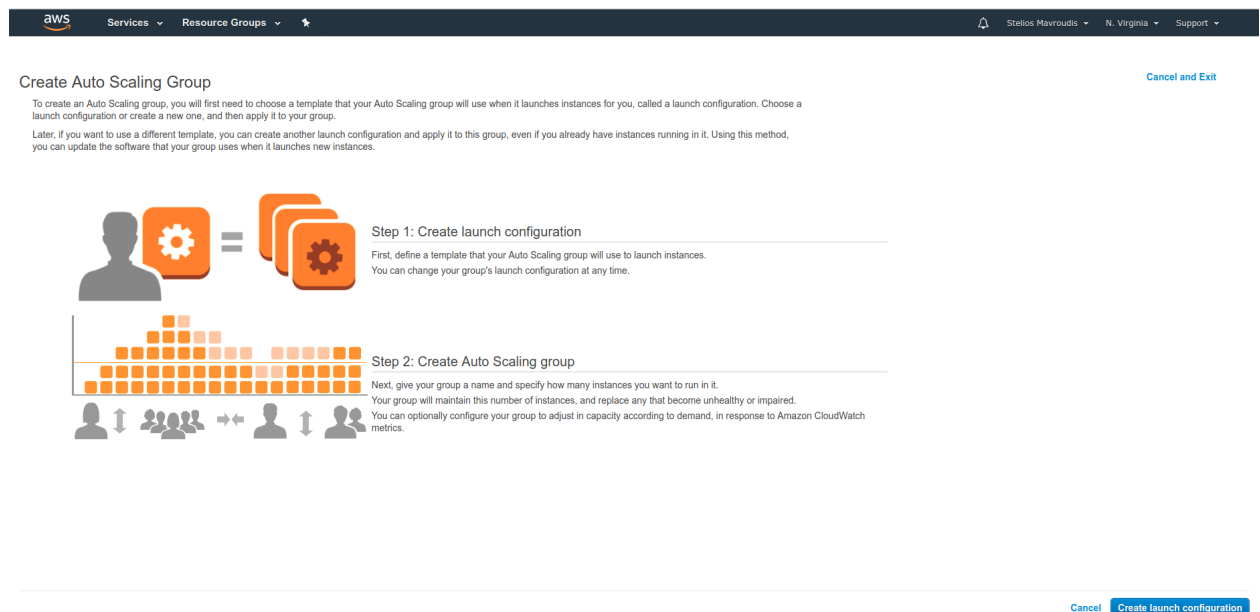
Εικόνα 27 Έλεγχος στοιχείων του χρήστη συστήματος

6.5 Δημιουργία εικονικών μηχανών

Έχοντας αναλύσει σε προηγούμενη ενότητα για εικονικά μηχανήματα EC2 της Amazon, παρακάτω φαίνεται πως χρησιμοποιήθηκαν οι οντότητες Auto Scaling Group και Launch Configuration της Amazon για να ορίσουμε τον τύπο μηχανημάτων αλλά και το μέγεθος της συστοιχίας υπολογιστών μας (cluster).

Auto Scaling Group: Ομάδα στην οποία ορίζουμε τύπου μηχανημάτων με κοινούς κανόνες αυτόματης κλιμάκωσης.

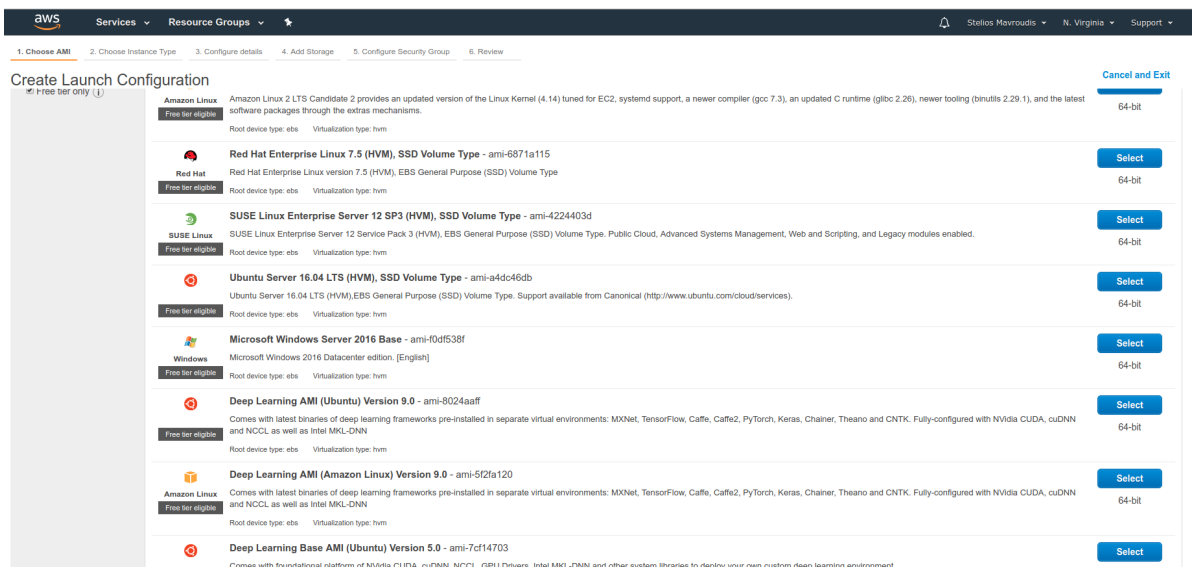
Launch Configuration: Ορίζει τον τύπο μηχανημάτων που θα μπει στην ομάδα αυτόματης κλιμάκωσης.



Εικόνα 28 Δημιουργία Auto Scaling Group βήμα 1

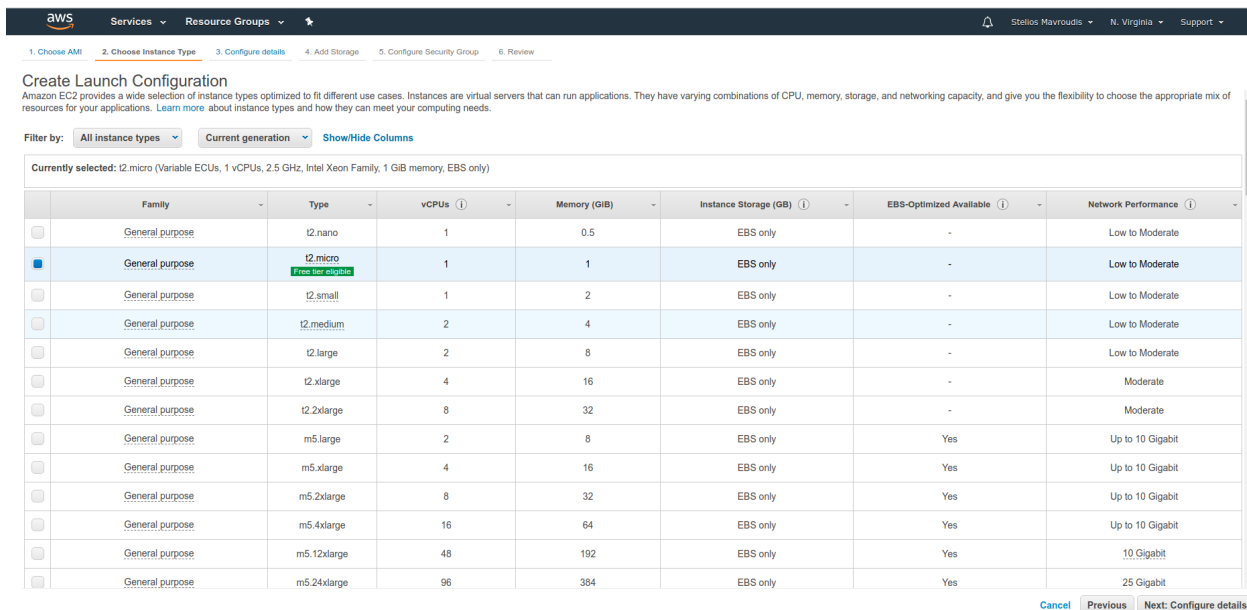
Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Στο πρώτο βήμα της δημιουργίας του **Launch Configuration** διαλέγουμε τον **τύπο** της εικονικής μηχανής. Στο οικοσύστημα της AWS τα εικονικά μηχανήματα ονομάζονται AMI (Amazon Machine Images).



Εικόνα 29 Επιλογή τύπου εικονικής μηχανής

Στο επόμενο βήμα διαλέγουμε το μέγεθος της εικονικής μηχανής.



Εικόνα 30 Επιλογή χαρακτηριστικών εικονικής μηχανής

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Στο επόμενο βήμα ορίζουμε το όνομα του **Launch Configuration (tei-pir-api)** και τον χρήστη συστήματος με τα κατάλληλα δικαιώματα για να δημιουργεί εικονικές μηχανές.

The screenshot shows the 'Create Launch Configuration' page in the AWS console, specifically the 'Configure details' step. The page has a dark header with the AWS logo and navigation menus. Below the header, there are six numbered steps: 1. Choose AMI, 2. Choose Instance Type, 3. Configure details (active), 4. Add Storage, 5. Configure Security Group, and 6. Review. The main content area is titled 'Create Launch Configuration' and contains several form fields: 'Name' with the value 'tei-pir-api', 'Purchasing option' with 'Request Spot Instances' unchecked, 'IAM role' with a dropdown menu showing 'ec2full', and 'Monitoring' with 'Enable CloudWatch detailed monitoring' unchecked. Below these fields is an 'Advanced Details' section with a message: 'Later, if you want to use a different launch configuration, you can create a new one and apply it to any Auto Scaling group. Existing launch configurations cannot be edited.' At the bottom right, there are navigation buttons: 'Cancel', 'Previous', 'Skip to review' (highlighted in blue), and 'Next: Add Storage'.

Εικόνα 31 Δημιουργία Launch Configuration Βήμα 1

Πριν το τελευταίο βήμα ορίζουμε την χωρητικότητα του εικονικού δίσκου που θα έχει η κάθε εικονική μηχανή.

The screenshot shows the 'Create Launch Configuration' page in the AWS console, specifically the 'Add Storage' step. The page has a dark header with the AWS logo and navigation menus. Below the header, there are six numbered steps: 1. Choose AMI, 2. Choose Instance Type, 3. Configure details, 4. Add Storage (active), 5. Configure Security Group, and 6. Review. The main content area is titled 'Create Launch Configuration' and contains a message: 'Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. https://docs.aws.amazon.com/console/ec2/launchinstance/storage about storage options in Amazon EC2.' Below this message is a table with columns: 'Volume Type', 'Device', 'Snapshot', 'Size (GiB)', 'Volume Type', 'IOPS', 'Throughput', 'Delete on Termination', and 'Encrypted'. The table has one row for the 'Root' volume with the following values: '/dev/sda1', 'snap-0eea1ed47e20333b8', '8', 'General Purpose (SSD)', '100 / 3000', 'N/A', checked, and 'No'. Below the table is an 'Add New Volume' button. At the bottom left, there is a message: 'Free tier eligible customers can get up to 30 GB of EBS storage. Learn more about free usage tier eligibility and usage restrictions.' At the bottom right, there are navigation buttons: 'Cancel', 'Previous', 'Skip to review' (highlighted in blue), and 'Next: Configure Security Group'.

Εικόνα 32 Δημιουργία Launch Configuration βήμα 2

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

1. Choose AMI 2. Choose Instance Type 3. Configure details 4. Add Storage 5. Configure Security Group 6. Review

Create Launch Configuration

Review the details of your launch configuration. You can go back to edit the details of each section before you finish.

⚠️ Improve security of instances launched using your launch configuration, tel-pir-api. Your security group, AutoScaling-Security-Group-1, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

AMI Details [Edit AMI](#)

Free tier eligible
Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-a4dc46db
Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).
Root device type: ebs Virtualization Type: hvm

Instance Type [Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory GiB	Instance Storage (GiB) GiB	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Launch configuration details [Edit details](#)

Name: tel-pir-api
Purchasing option: On demand
EBS Optimized: No
Monitoring: No
IAM role: ec2Full
Tenancy: Shared tenancy (multi-tenant hardware)
Kernel ID: Use default
RAM Disk ID: Use default
User data:
IP Address Type: Only assign a public IP address to instances launched in the default VPC and subnet. (default)

Cancel Previous **Create launch configuration**

Εικόνα 33 Δημιουργία Launch Configuration βήμα 2

Για την ασφαλή απομακρυσμένη σύνδεση στα εικονικά μηχανήματα δημιουργούμε ένα κρυπτογραφημένο αρχείο **pem** που θα είναι το κλειδί μας για την επικοινωνία με το cluster.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair

Select a key pair
deployer

I acknowledge that I have access to the selected private key file (deployer.pem), and that without this file, I won't be able to log into my instance.

Cancel **Create launch configuration**

Εικόνα 34 Χρήση κρυπτογραφημένου αρχείου pem

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Αφού δημιουργήσαμε το **Launch Configuration** ορίζουμε τις παραμέτρους του νέου **Auto Scaling Group** με όνομα **tei-pir-api-asg** και με αριθμό μηχανημάτων δύο.

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Create Auto Scaling Group Cancel and Exit

Launch Configuration ⓘ tei-pir-api

Group name ⓘ tei-pir-api-asg

Group size ⓘ Start with instances

Network ⓘ vpc-ee96b597 (172.31.0.0/16) (default) ⊞ Create new VPC

Subnet ⓘ subnet-09470d25 (172.31.0.0/20) | Default in us-east-1 ⊞
subnet-9375ba7f (172.31.0.0/20) | Default in us-east-1b ⊞ ⊞ Create new subnet

Each instance in this Auto Scaling group will be assigned a public IP address. ⓘ

▶ Advanced Details

Cancel Next: Configure scaling policies

Εικόνα 35 Δημιουργία Auto Scaling Group Βήμα 1

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Create Auto Scaling Group

Please review your Auto Scaling group details. You can go back to edit changes for each section. Click **Create Auto Scaling group** to complete the creation of an Auto Scaling group.

▼ Auto Scaling Group Details Edit details

Group name tei-pir-api-asg

Group size 2

Minimum Group Size 2

Maximum Group Size 2

Subnet(s) subnet-9375ba7f, subnet-09470d25

Health Check Grace Period 300

Detailed Monitoring No

Instance Protection None

Service-Linked Role AWSServiceRoleForAutoScaling

▼ Scaling Policies Edit scaling policies

▼ Notifications Edit notifications

▼ Tags Edit tags

Cancel Previous Create Auto Scaling group

Εικόνα 36 Δημιουργία Auto Scaling Group βήμα 2

Auto Scaling group creation status

✔ Successfully created Auto Scaling group View creation log

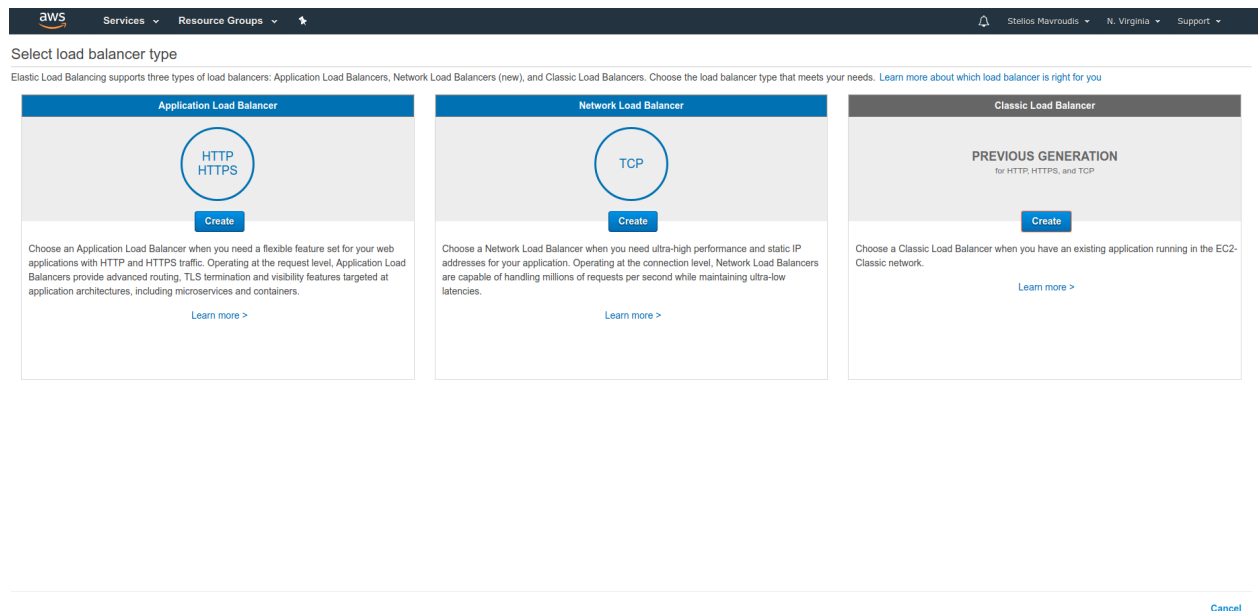
• View View your Auto Scaling groups
View your launch configurations

• Here are some helpful resources to get you started

Close

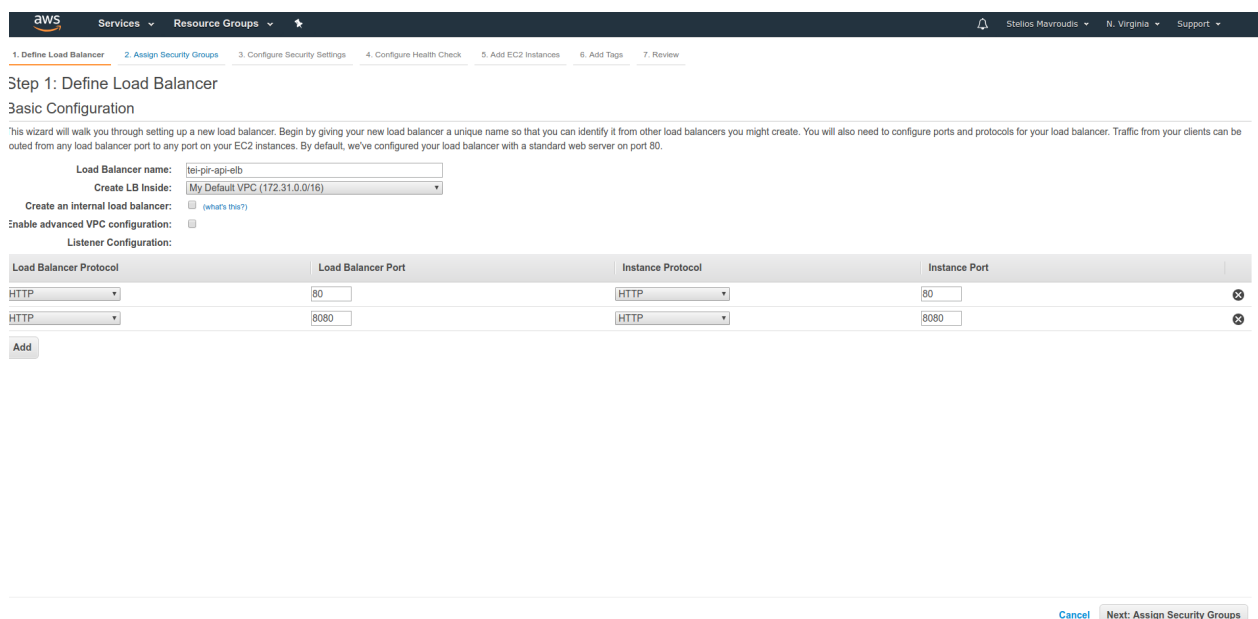
6.6 Δημιουργία κατανεμητή φορτίου (Load Balancer)

Αφού το cluster μας θα έχει τουλάχιστον δυο μηχανήματα να σερβίρουν την εφαρμογή μας θα χρειαστεί να δημιουργήσουμε έναν κατανεμητή φορτίου (Load Balancer). Ο κατανεμητής φορτίου μοιράζει τις αιτήσεις προς την εφαρμογή ισότιμα ώστε τα μηχανήματα να παραμένουν υγιή.



Εικόνα 37 Δημιουργία κατανεμητή φορτίου βήμα 1

Ο Load Balancer βοηθάει επίσης να ορίσουμε με ασφάλεια ποιες διαδικτυακές πόρτες είναι ανοιχτές προς την εφαρμογή. Στην περίπτωση της εφαρμογής της διπλωματικής θα πρέπει να αφήσουμε την διαδικτυακή κίνηση να περνάει από την δικτυακή πόρτα 8080.



Εικόνα 38 Δημιουργία κατανεμητή φορτίου βήμα 2

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Step 2: Assign Security Groups

You have selected the option of having your Elastic Load Balancer inside of a VPC, which allows you to assign security groups to your load balancer. Please select the security groups to assign to this load balancer. This can be changed at any time.

Assign a security group: Create a new security group Select an existing security group

Security Group ID	Name	Description	Actions
sg-8aadec2	AutoScaling-Security-Group-1	AutoScaling-Security-Group-1 (2018-05-27 19:36:42.007+03:00)	Copy to new
sg-e2f96992	default	default VPC security group	Copy to new
sg-b582c3fd	launch-wizard-1	launch-wizard-1 created 2018-05-27T14:34:41.017+03:00	Copy to new
sg-985110d0	rds-launch-wizard	Created from the RDS Management Console: 2018/05/27 10:50:15	Copy to new

Filter VPC security groups

Cancel Previous Next: Configure Security Settings

Εικόνα 39 Δημιουργία καταναμητή φορτίου βήμα 3

Στο Load Blancer θα πρέπει να του ορίσουμε έναν σύνδεσμο της εφαρμογής στο οποίο θα ελέγχει ότι το σύστημα είναι υγιές. Έτσι αν αυτό δεν ισχύει να κόβει την κίνηση προς το συγκεκριμένο μηχάνημα που έχει το πρόβλημα.

Step 4: Configure Health Check

Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.

Ping Protocol: HTTP
Ping Port: 80
Ping Path: /index.html

Advanced Details

Response Timeout: 5 seconds
Interval: 30 seconds
Unhealthy threshold: 2
Healthy threshold: 10

Cancel Previous Next: Add EC2 Instances

Εικόνα 40 Δημιουργία καταναμητή φορτίου βήμα 4

Τέλος αντιστοιχούμε στον καταναμητή φορτίου τα μηχανήματα που σερβίρουν την εφαρμογή μας.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

aws Services Resource Groups

1. Define Load Balancer 2. Assign Security Groups 3. Configure Security Settings 4. Configure Health Check 5. Add EC2 Instances 6. Add Tags 7. Review

Step 5: Add EC2 Instances

The table below lists all your running EC2 Instances. Check the boxes in the Select column to add those instances to this load balancer.

VPC vpc-ee96b597 (172.31.0.0/16)

Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
<input type="checkbox"/>	i-0bb557761b8107252	running	AutoScaling-Security-Group-1	us-east-1c	subnet-09470d25	172.31.80.0/20
<input type="checkbox"/>	i-0da3e7a9eb1f04ee6	running	AutoScaling-Security-Group-1	us-east-1b	subnet-9375ba7f	172.31.0.0/20

Availability Zone Distribution
1 instance in us-east-1b
1 instance in us-east-1c

Enable Cross-Zone Load Balancing ⓘ
 Enable Connection Draining ⓘ 300 seconds

Cancel Previous Next: Add Tags

Εικόνα 41 Δημιουργία κατανεμητή φορτίου βήμα 5

aws Services Resource Groups

1. Define Load Balancer 2. Assign Security Groups 3. Configure Security Settings 4. Configure Health Check 5. Add EC2 Instances 6. Add Tags 7. Review

Step 7: Review

Please review the load balancer details before continuing.

Define Load Balancer [Edit load balancer definition](#)

Load balancer name: tei-pir-api-elb
Scheme: internet-facing
Port Configuration: 80 (HTTP) forwarding to 80 (HTTP)
8080 (HTTP) forwarding to 8080 (HTTP)

Configure Health Check [Edit health check](#)

Ping Target: HTTP:80/index.html
Timeout: 5 seconds
Interval: 30 seconds
Unhealthy threshold: 2
Healthy threshold: 10

Add EC2 Instances [Edit instances](#)

Cross-Zone Load Balancing: Enabled
Connection Draining: Enabled, 300 seconds
Instances: i-0bb557761b8107252, i-0da3e7a9eb1f04ee6

VPC Information [Edit subnets](#)

VPC: vpc-ee96b597
Subnets: subnet-9375ba7f, subnet-2e0e4222, subnet-01d53a3e, subnet-2631386e, subnet-09470d25, subnet-8c8cc0d6

Security groups [Edit security groups](#)

Security groups: sg-e2966992, sg-8aadcd2

Cancel Previous Create

Load Balancer Creation Status

Successfully created load balancer
Load balancer `tei-pir-api-elb` was successfully created.
Note: It may take a few minutes for your instances to become active in the new load balancer.

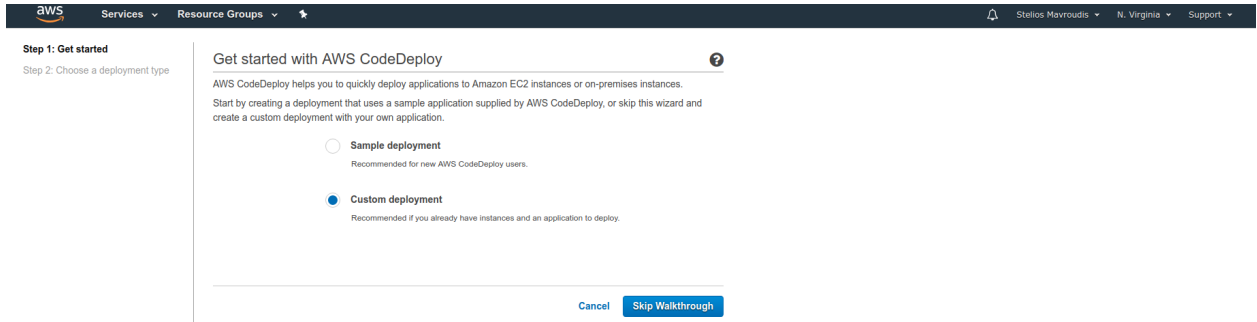
Close

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

6.7 Παραμετροποίηση της υπηρεσίας CodeDeploy του AWS

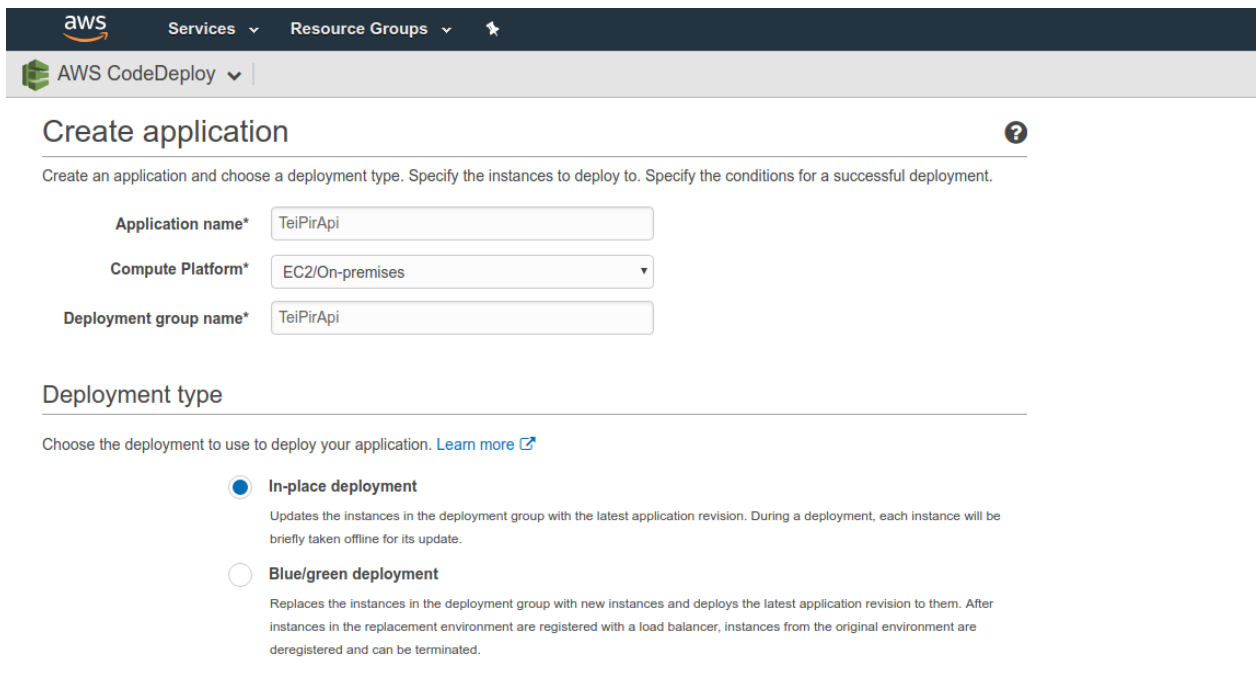
Στα προηγούμενα βήματα δημιουργήσαμε και παραμετροποιήσαμε όλες τις απαραίτητες οντότητες για την δημιουργία της αυτοματοποιημένης διαδικασίας ανανέωσης της έκδοσης της εφαρμογής μας στο Cloud (Automation Deploy). [12]

Έτσι παρακάτω φαίνεται βήμα βήμα η τελευταία διαδικασία.



Εικόνα 42 Παραμετροποίηση της υπηρεσίας CodeDeploy βήμα 1

Το όνομα αυτής της αυτοματοποιημένης διαδικασίας που θα δημιουργήσουμε είναι **TeiPirApi** και θα χρησιμοποιεί τα ίδια μηχανήματα για να αντικαταστήσει την παλιά έκδοση της εφαρμογής μας με την καινούρια (In-place deployment) και όχι αυτή της αντικατάστασης μηχανημάτων (Blue/Green deployment).



Εικόνα 43 Παραμετροποίηση υπηρεσίας CodeDeploy βήμα 2

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Στο επόμενο βήμα του ορίζουμε να χρησιμοποιεί τα μηχανήματα από το Auto Scaling Group που δημιουργήσαμε σε προηγούμενο βήμα και έτσι εμφανίζονται παρακάτω και τα δυο υπάρχων μηχανήματα.

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instances to this deployment group.

Auto Scaling groups **Amazon EC2 instances** **On-premises instances**

You can select up to 10 Auto Scaling groups to deploy your application revision to.

	Name	Instances	
1	tei-pir-api-asg	2	✕
2			✕

Matching instances

Right now these instances match the criteria you specified. There might be more or fewer instances that match your criteria when a deployment runs.

« < 1 to 2 of 2 instances > »

Instance ID	Status	Filter types	Association
i-0bb557761b8107252	Healthy	asg name:tei-pir-api-asg	ASG
i-0da3e7a9eb1f04ee6	Healthy	asg name:tei-pir-api-asg	ASG

Εικόνα 44 Παραμετροποίηση υπηρεσίας CodeDeploy βήμα 3

Αφού ορίσαμε ποια μηχανήματα θα χρησιμοποιεί θα πρέπει να δηλώσουμε και ποιος είναι ο αντίστοιχος κατανομητής φορτίου για τα αντίστοιχα μηχανήματα. Έτσι ορίζουμε τον **tei-pir-api-elb**.

Enable load balancing

Load balancer Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed to and allows traffic to it again after the deployment succeeds.

Application Load Balancer or Network Load Balancer

Classic Load Balancer

tei-pir-api-elb

Deployment configuration

Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application will be deployed and the success or failure conditions for a deployment.

Deployment configuration CodeDeployDefault.OneAtATime or **Create deployment configuration**

Routes traffic to one instance in the replacement environment at a time. Succeeds if traffic is successfully rerouted to all replacement instances. Fails after the very first rerouting failure. Allows the deployment to succeed for some instances, even if the overall deployment fails.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

Ο Load Balancer κατά την διάρκεια που ένα μηχάνημα ανανεώνεται με την καινούρια έκδοση της εφαρμογής μας σταματάει την κίνηση σε αυτό το μηχάνημα μέχρι αυτό να είναι έτοιμο ώστε να συνεχίσει με την ανανέωση του επόμενου μηχανήματος.

Σαν τελικό βήμα ορίζουμε το χρήστη συστήματος με τα αντίστοιχα δικαιώματα για την πραγματοποίηση της αυτόματης διαδικασίας ανανέωση της έκδοσης της εφαρμογής μας στα μηχανήματα.

Service role

Select a service role that grants AWS CodeDeploy access to the instances.

Service role ARN*

*Required

Cancel

Create application

Εικόνα 45 Παραμετροποίηση υπηρεσίας CodeDeploy βήμα 4

The screenshot shows the AWS CodeDeploy console interface. At the top, there's a navigation bar with 'AWS CodeDeploy' and 'Applications > TeiPirApi'. A green success message states: 'Congratulations! The application TeiPirApi has been created.' Below this, there are instructions and a terminal command: `aws deploy push --application-name <MyAppName> \ --s3-location s3://<MyBucketName>/<MyNewAppBundleName> \ --source <PathToMyBundle>`. The 'Application details' section shows 'Name: TeiPirApi' and 'Compute platform: EC2/On-premises'. The 'Deployment groups' section has a 'Create deployment group' button and a table with one entry: 'TeiPirApi'.

Name	Status	Last attempted revision	Last successful revision	Triggers
TeiPirApi				

6.8 Σύνδεση Git με την υπηρεσία CodeDeploy της Amazon

Το Git είναι ένα σύστημα ελέγχου εκδόσεων (λέγεται και σύστημα ελέγχου αναθεωρήσεων ή σύστημα ελέγχου πηγαίου κώδικα) με έμφαση στην ταχύτητα, στην ακεραιότητα των δεδομένων και στην υποστήριξη για καταναμημένες μη γραμμικές ροές εργασίας. [18]

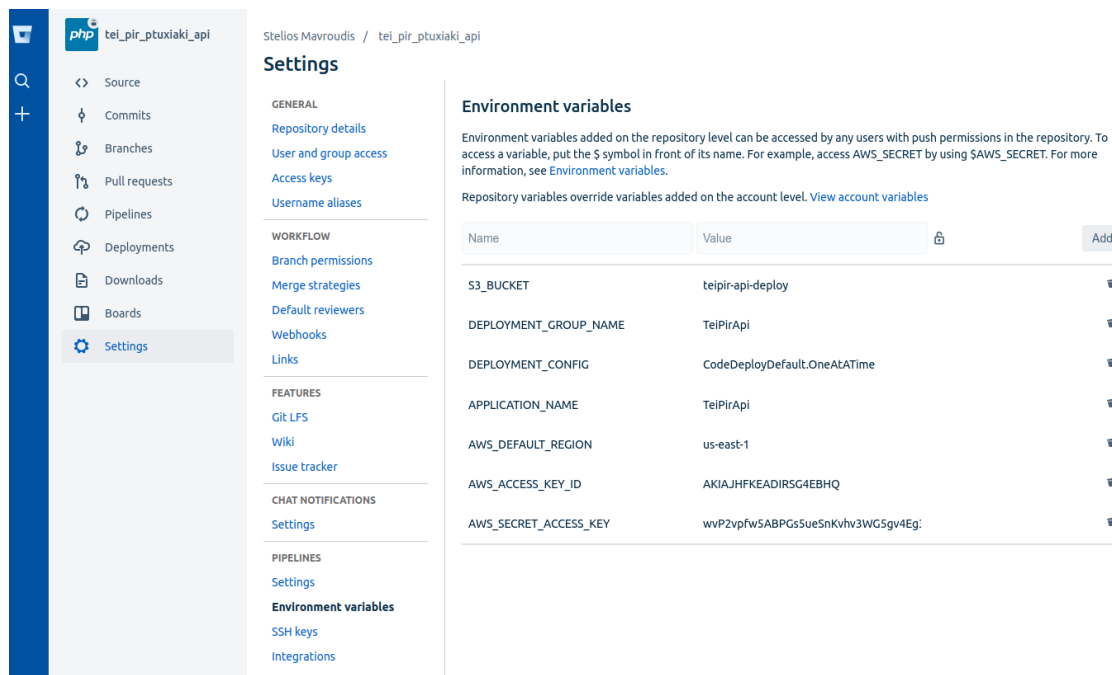
Το Git σχεδιάστηκε και αναπτύχθηκε αρχικά από τον Λίνους Τόρβαλντς για τη ανάπτυξη του πυρήνα Linux το 2005 και έχει γίνει από τότε το πιο διαδεδομένο σύστημα ελέγχου εκδόσεων για ανάπτυξη λογισμικού.

Το Git έχει αποθετήρια κώδικα (Repositories) και το κάθε αποθετήριο κώδικα έχει κλαδιά εκδόσεων κώδικα (Branches).

Υπάρχουν διάφορες διαδικτυακές πλατφόρμες για την παροχή της υπηρεσίας και την διαχείριση του git (Github, Bitbucket, Gitlab και άλλες).

Για την εφαρμογή μας χρησιμοποιήθηκε το Bitbucket.

Στην παρακάτω εικόνα φαίνεται πώς προσθέσαμε για το έργο (Project) μεταβλητές περιβάλλοντος (Environment Variables) για να μπορέσει να γίνει η διασύνδεση μεταξύ της υπηρεσίας CodeDeploy με το Bitbucket.



The screenshot shows the Bitbucket 'Settings' page for a repository named 'tei_pir_ptuxiaki_api'. The 'Environment variables' section is active, displaying a table of variables:

Name	Value	Visibility	Action
S3_BUCKET	teipir-api-deploy	Public	Down Arrow
DEPLOYMENT_GROUP_NAME	TeiPirApi	Public	Down Arrow
DEPLOYMENT_CONFIG	CodeDeployDefault.OneAtATime	Public	Down Arrow
APPLICATION_NAME	TeiPirApi	Public	Down Arrow
AWS_DEFAULT_REGION	us-east-1	Public	Down Arrow
AWS_ACCESS_KEY_ID	AKIAJHFKEADIRSG4EBHQ	Private	Down Arrow
AWS_SECRET_ACCESS_KEY	wvP2vpfw5ABPGs5ueSnKvhv3WG5gv4Eg:	Private	Down Arrow

Εικόνα 46 Μεταβλητές περιβάλλοντος για την διασύνδεση Bitbucket με CodeDeploy

Εκτός από τις μεταβλητές περιβάλλοντος θα πρέπει να δημιουργήσουμε και ορισμένα αρχεία στην εφαρμογή μας που θα περιγράφουν το πώς θα γίνει η διαδικασία του deployment αλλά και τι διαδικασίες πρέπει να γίνουν πριν και μετά το deployment.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

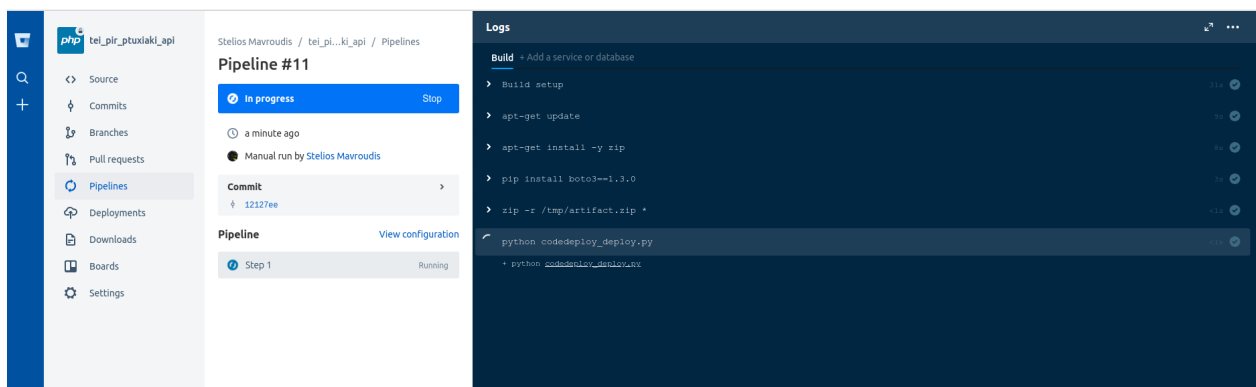
Το `appspec.yml` είναι το αρχείο που ορίζει ποια αρχεία θα αποθηκευτούν που στα μηχανήματα και ποια είναι τα σκριπτ που θα τρέξουν κατά την διάρκεια του deployment όπως για παράδειγμα πριν την εγκατάσταση της εφαρμογής να γίνει εγκατάσταση στα μηχανήματα των απαραίτητων εφαρμογών ή βιβλιοθηκών.

```
version: 0.0
os: linux
files:
  - source: /*
    destination: /var/www/html/
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies
      timeout: 300
      runas: root
    - location: scripts/start_server
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server
      timeout: 300
      runas: root
```

Ο φάκελος `scripts` περιέχει όλες τα script αρχεία με τις απαραίτητες εντολές συστήματος για την ανάλογη κατάσταση.

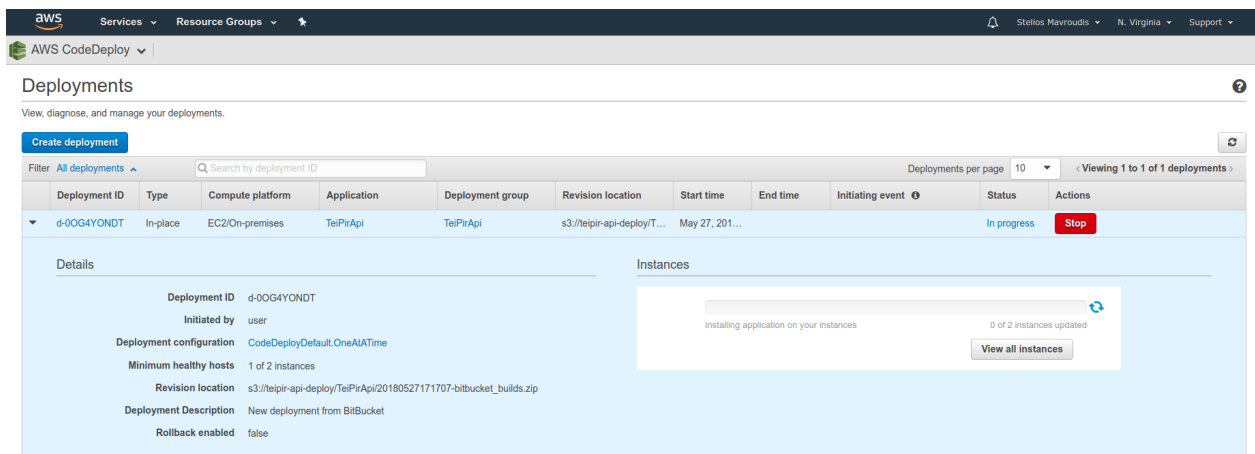
Τα αρχεία `bitbucket-pipelines.yml` και `codedeploy_deploy.py` είναι αρχεία πρότυπα (templates) και τα χρησιμοποιεί το Bitbucket.

Έτσι μετά από τα παραπάνω βήματα θα μπορούμε να πάμε στην ενότητα Pipelines του Bitbucket και να τρέξουμε ένα αυτοματοποιημένο deployment και αντίστοιχα να το δούμε την εξέλιξή του και μέσα από την διαδικτυακή κονσόλα της Amazon (AWS web console).



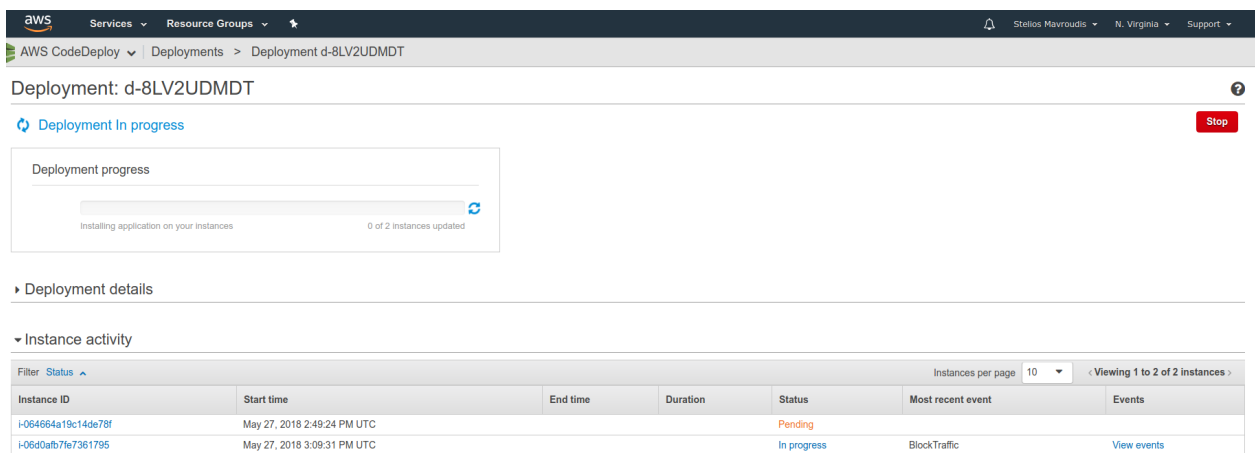
Εικόνα 47 Deployment σε εξέλιξη από την πλατφόρμα του Bitbucket

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.



The screenshot shows the AWS CodeDeploy console. At the top, there's a navigation bar with 'AWS', 'Services', and 'Resource Groups'. Below that, the 'AWS CodeDeploy' breadcrumb is visible. The main heading is 'Deployments'. A 'Create deployment' button is on the left. A table lists deployments, with one selected: 'd-00G4YONDT' (In-place, EC2/On-premises, TeiPirApi, TeiPirApi, s3://teipir-api-deploy/T..., May 27, 201..., In progress, Stop). Below the table, the 'Details' section shows: Deployment ID: d-00G4YONDT, Initiated by: user, Deployment configuration: CodeDeployDefault.OneAtATime, Minimum healthy hosts: 1 of 2 instances, Revision location: s3://teipir-api-deploy/TeiPirApi/20180527171707-bitbucket_builds.zip, Deployment Description: New deployment from BitBucket, Rollback enabled: false. The 'Instances' section shows a progress bar for 'Installing application on your instances' with '0 of 2 instances updated' and a 'View all instances' button.

Εικόνα 48 Deployment σε εξέλιξη από την AWS κονσόλα



The screenshot shows the 'Deployment: d-8LV2UDMDT' page in the AWS CodeDeploy console. It includes a 'Deployment In progress' status with a 'Stop' button. A 'Deployment progress' section shows a progress bar for 'Installing application on your instances' with '0 of 2 instances updated'. Below this, there are sections for 'Deployment details' and 'Instance activity'. The 'Instance activity' table lists two instances:

Instance ID	Start time	End time	Duration	Status	Most recent event	Events
i-06466419c14de78f	May 27, 2018 2:49:24 PM UTC			Pending		
i-06d0afb7e7361795	May 27, 2018 3:09:31 PM UTC			In progress	BlockTraffic	View events

ΣΥΜΠΕΡΑΣΜΑΤΑ

Με την συγκεκριμένη διπλωματική εργασία καταφέραμε να αναπτύξουμε μια εφαρμογή συνδυάζοντας σύγχρονες τεχνολογίες με στόχο να είναι σταθερή και εύκολα επεκτάσιμη, τόσο ή ίδια η εφαρμογή αλλά και η υποδομή που την υποστηρίζει.

Χρησιμοποιώντας ένα σύγχρονο και πάνοπλο από λειτουργίες framework της επιλεγμένης γλώσσας προγραμματισμού για την ανάπτυξη της εφαρμογής μας καταφέρνουμε να έχουμε μια καλά δομημένη αρχιτεκτονική με σκοπό την ταχύτερη ανάπτυξη σε συνδυασμό με τα λιγότερα λάθη.

Για την συγκεκριμένη εφαρμογή το framework ήταν το Symfony της γλώσσας PHP, αλλά θα μπορούσε να ήταν και ο συνδυασμός Spring με την γλώσσα προγραμματισμού Java, αποτελώντας έτσι τη χρήση framework ως βασική φιλοσοφία για την έναρξη και σχεδίαση μιας εφαρμογής.

Η εικονοποίηση υλικού και λειτουργικών συστημάτων βοηθά στην κατάργηση του φόβου για τον όγκο πληροφορίας που θα έχει να επεξεργαστεί μια εφαρμογή καθώς επίσης και η δυναμικότητα στην κλιμάκωση μειώνει τα κόστη υποστήριξης της λειτουργίας μιας εφαρμογής.

Η μεταφερισιμότητα και η χρήση ενός εικονικού λογισμικού όχι μόνο μειώνει την ύπαρξη λαθών λόγω της διαφορετικότητας του συστήματος που υπήρχε παλιότερα, αλλά βοηθά και στην γρηγορότερη επίλυση αυτών αν υπάρχουν, λόγω της κοινής συμπεριφοράς του συστήματος.

Γράφοντας σενάρια τεστ για την λογική λειτουργίας της εφαρμογής αποκτάμε σταθερότητα αλλά και προλαβαίνουμε στις περισσότερες φορές το λάθος να πάει στον τελικό χρήστη. Με τον συνδυασμό τεστ σεναρίων μαζί με την μεθοδολογία Continuous Integration πετυχαίνουμε την ταχύτερη ανανέωση έκδοσης της εφαρμογής μας προς τον πελάτη και τον τελικό χρήστη.

Ανάπτυξη REST API για mobile e-banking εφαρμογές με την χρήση νέων τεχνολογιών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] Sohail Salehi Mastering Symfony (April 2016)
- [2] Symfonny The Best Practices Book October 24, (2017)
- [3] Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley, Dan Mackin UNIX and Linux System Administration Handbook (5th Edition)
- [4] Pethuru Raj, Jeeva S. Chelladurai, Vinod Singh Learning Docker (2015)
- [5] Clément Nedelcu Nginx HTTP Server, 2nd Edition (2013)
- [6] Leonard Richardson and Sam Ruby RESTful Web Services (2007)
- [7] Charles Bell, Mats Kindahl, and Lars Thalmann MySQL High Availability (2014)
- [8] PHP-FPM, Retrieved from <http://php.net/manual/en/install.fpm.php>
- [9] RDS, Retrieved from <https://aws.amazon.com/rds/>
- [10] EC2. Retrieved from <https://aws.amazon.com/ec2/>
- [11] S3, Retrieved from <https://aws.amazon.com/s3/>
- [12] CodeDeploy, Retrieved from <https://aws.amazon.com/codedeploy/>
- [13] JWT, Retrieved from <https://jwt.io/>
- [14] Virtualization , Retrieved from <https://en.wikipedia.org/wiki/Virtualization>
- [15] Symfony generate controller, Retrieved from https://symfony.com/doc/current/bundles/SensioGeneratorBundle/commands/generate_controller.html
- [16] Symfony generate entity, Retrieved from <https://symfony.com/doc/master/bundles/SensioGeneratorBundle/commands/generateDoctrineEntity.html>
- [17] Symfony generate bundles, Retrieved from <https://symfony.com/doc/master/bundles/SensioGeneratorBundle/commands/generateBundle.html>
- [18] Git, Retrieved from <https://en.wikipedia.org/wiki/Git>
- [19] Doctrine ORM, Retrieved from <https://symfony.com/doc/current/doctrine.html>