

Κατασκευή Έξυπνης, Τηλεχειριζόμενης  
Κλειδαριάς Θυροτηλεφώνου με χρήση  
Τεχνολογιών Αιχμής

Πανεπιστήμιο Πειραιώς

Κυριάκος Δ. Γιαννάκης

Day Τεστ Year

## **Abstract**

TODO

# Contents

<b>1 Εισαγωγή</b>	<b>4</b>
1.1 Internet of Things . . . . .	4
1.2 Αυτοματισμοί Σπιτιού - Home Automation . . . . .	5
1.3 Σκοπός του PiLock . . . . .	5
1.4 Έρευνα αγοράς - Καινοτομία του PiLock . . . . .	5
1.4.1 Μη Εμπορικές/DIY Εφαρμογές . . . . .	5
1.4.2 Εμπορικές Εφαρμογές . . . . .	6
1.4.3 Διαφορές του PiLock με τις ήδη υπάρχουσες εφαρμογές . . . . .	6
<b>2 Δομή του PiLock</b>	<b>7</b>
2.1 Σύντομη Περιγραφή Λογισμικού Εξυπηρετητή - PiLock Server . . . . .	7
2.2 Σύντομη Περιγραφή Λογισμικού Πελάτη - PiLock Client . . . . .	7
2.3 Υλικό - Hardware . . . . .	8
2.3.1 Raspberry Pi Zero W . . . . .	8
2.3.2 Relay Module . . . . .	9
2.3.3 Arduino UNO . . . . .	9
2.3.4 Λοιπό Hardware . . . . .	10
2.4 Κόστος Κατασκευής του PiLock . . . . .	11
<b>3 Συστήματα Ελέγχου Πρόσβασης Πολυκατοικιών/Σπιτιών</b>	<b>13</b>
<b>4 Νέος μηχανισμός ξεκλειδώματος</b>	<b>16</b>
4.1 Προετοιμασία του Raspberry Pi . . . . .	16
4.1.1 Άλλαγή των προεπιλεγμένων στοιχείων πρόσβασης . . . . .	17
4.1.2 Ενημέρωση του Raspbian . . . . .	17
4.1.3 Ανάθεση στατικής διεύθυνσης IP . . . . .	17
4.2 Σύνδεση με το Relay . . . . .	18
4.2.1 Σύνδεση χωρίς την χρήση Arduino . . . . .	18
4.2.2 Σύνδεση με την χρήση Arduino . . . . .	18
<b>5 Προγραμματιστικό Περιβάλλον - Τεχνολογίες που Χρησιμοποιήθηκαν</b>	<b>21</b>
5.1 Η σημασία χρήσης δωρεάν λογισμικού ανοικτού κώδικα κατά την ανάπτυξη του PiLock . . . . .	21
5.2 Άδεια Χρήσης του PiLock . . . . .	22
5.3 Διαχείρηση του έργου . . . . .	22
5.3.1 Version Control . . . . .	22
5.3.2 Issue Tracking . . . . .	23
5.4 Προγραμματιστικό Περιβάλλον . . . . .	24
5.4.1 Γλώσσες Προγραμματισμού/Markup . . . . .	24

5.4.2	Βιβλιοθήκες/Frameworks που χρησιμοποιήθηκαν . . . . .	24
5.4.3	Προγραμματιστικά Εργαλεία που Χρησιμοποιήθηκαν . . . . .	25
<b>6</b>	<b>Χρονοδιάγραμμα Κυκλοφορίας Εκδόσεων</b>	<b>27</b>
6.1	0.1.0 . . . . .	27
6.2	0.2.0 . . . . .	28
6.3	0.3.0 . . . . .	28
6.4	0.3.1 . . . . .	28
<b>7</b>	<b>Σενάρια Ξεκλειδώματος - PiLockUnlockScripts</b>	<b>30</b>
7.1	Ξεκλειδωμα μέσω GPIO . . . . .	31
7.2	Ξεκλειδωμα μέσω Arduino . . . . .	31
7.3	Χρήση των σεναρίων ξεκλειδώματος . . . . .	32
<b>8</b>	<b>Ο Διακομιστής του PiLock - PiLock Server</b>	<b>34</b>
8.1	Αρχιτεκτονική MTV . . . . .	34
8.2	Django Apps . . . . .	35
8.3	Μοντέλα που Ορίστηκαν/Χρησιμοποιούνται . . . . .	35
8.4	Σύστημα Εξουσιοδότησης Πρόσβασης . . . . .	36
8.4.1	Στάδιο Εισόδου - Login . . . . .	37
8.4.2	Στάδιο Ξεκλειδώματος - Unlock . . . . .	38
8.4.3	Ξεκλειδωμα μέσω Android Wear - Wear Unlock . . . . .	39
8.4.4	Αλλαγή PIN χρηστών . . . . .	39
8.4.5	Ασφάλεια . . . . .	40
8.5	Λειτουργία Heartbeat . . . . .	41
8.6	Γενικός Διακόπτης . . . . .	41
<b>9</b>	<b>Η εφαρμογή Android - PiLock Client</b>	<b>44</b>
9.1	Activities . . . . .	44
9.2	Αιτήματα HTTPS . . . . .	45
9.2.1	Το πρόβλημα με τα Αυτο-Υπογεγραμμένα Πιστοποιητικά SSL .	46
9.3	Μηχανισμός Heartbeat . . . . .	46
9.4	Κρυπτογράφηση Τεκμηρίων Πρόσβασης . . . . .	47
<b>10</b>	<b>Η εφαρμογή Android Wear</b>	<b>49</b>
<b>11</b>	<b>Ο Πίνακας Διαχείρησης του PiLock</b>	<b>50</b>
11.1	Ημερολόγιο Πρόσβασης - Access Log . . . . .	51
11.2	Σύστημα ειδοποιήσεων . . . . .	51
<b>12</b>	<b>Εγκατάσταση και ρύθμιση του PiLock Server</b>	<b>53</b>
<b>13</b>	<b>Συνεχής Ενσωμάτωση - Continuous Integration</b>	<b>55</b>
<b>14</b>	<b>Μελλοντική Επέκταση του PiLock</b>	<b>57</b>
14.1	Σύνδεση με Κεντρικό Σημείο Ελέγχου Έξυπνων Συσκευών . . . . .	57
14.2	Ξεκλειδωμα μέσω αναγνώρισης δακτυλικού αποτυπώματος . . . . .	57
14.3	Ενσωμάτωση αυθεντικοίσης μέσω LDAP, AD DS . . . . .	58
14.4	Ξεκλειδωμα Επισκεπτών . . . . .	58

<b>15 Επίλογος - Συμπεράσματα</b>	<b>59</b>
<b>A Διαγράμματα</b>	<b>61</b>
A.1 Συνοπτικά Διαγράμμιατα Κλάσεων Εκδόσεων . . . . .	61
A.2 Διαγράμμιατα Περιπτώσεων Χρήσεων Εκδόσεων . . . . .	65

# Κεφάλαιο 1

## Εισαγωγή

Στον σημερινό κόσμο, οι τεχνολογικές μιας ανάγκες γίνονται ολοένα και πιο πολύπλοκες. Κάθε μέρα βγαίνουν στην επιφάνεια νέες τεχνολογικές διευκολύνσεις για τον άνθρωπο, σκοπός των οποίων είναι να κάνουν την διαβίωσή του πιο ”έξυπνη”, δίνοντάς του τον μέγιστο έλεγχο σε κάθε σημείο της ζωής του. Με την άνθιση του internet of things, γίνεται εύκολη η διασύνδεση πολλών συσκευών (από την μικρότερη ως την μεγαλύτερη), με σκοπό τον έλεγχό τους και την άντληση δεδομένων από αυτές, απομακρυσμένα.

**Σκοπός της παρούσας πτυχιακής εργασίας είναι να περιγράψει την πλήρη διαδικασία του σχεδιασμού και υλοποίησης ενός συστήματος ελέγχου κλειδαριάς σπιτιού/γραφείου, γνωστό ως PiLock.**

Η εφαρμογή υλοποιήθηκε, στο μεγαλύτερο μέρος της, χρησιμοποιώντας λογισμικό τελευταίας τεχνολογίας, πράγμα που μιας εγγυάται την μέγιστη ευελιξία όσων αφορά την ανάπτυξη, πράγμα που ισοδυναμεί με μέγιστη ταχύτητα ανάπτυξης και αυξημένη ασφάλεια.

### 1.1 Internet of Things

Ο όρος ”Internet of Things” (IoT) χρησιμοποιήθηκε πρώτη φορά από τον Kevin Ashton το 1999 σε μία παρουσίασή του στην Procter & Gamble (P&G) [1]. Ο όρος επινοήθηκε προκειμένου να μπορεί να τονιστεί η δύναμη της (τότε) δημοφιλούς ιδέας της χρήσης της τεχνολογίας RFID σε συστήματα εφοδιαστικών αλυσίδων εταιριών για παρακολούθηση εμπορευμάτων. Πλέον, ο όρος Internet of Things χρησιμοποιείται προκειμένου να χαρακτηριστούν συσκευές (μικρές ή μεγάλες) με δυνατότητα σύνδεσης στο Internet. Κάποια παραδείγματα είναι τα αυτοκίνητα με ενσωματομένους αισθητήρες, τα έξυπνα σπίτια (τα οποία αποτελούνται από μια πληθώρα έξυπνων συσκευών), καθώς επίσης και συγκεχριμένες συσκευές παρακολούθησης υγείας (όπως πχ. συσκευές παρακολούθησης καρδιακού ρυθμού) με δυνατότητα σύνδεσης στο διαδίκτυο.

Οι δυνατότητες που έχουν οι συγκεχριμένες συσκευές τις καθιστούν ικανές για σύνδεση στο internet, και κατ’επέκταση, αυξάνουν σημαντικά τις λειτουργίες τους, προσδιδόντας μεγαλύτερο έλεγχο στον χρήστη.

## 1.2 Αυτοματισμοί Σπιτιού - Home Automation

Μία από τις πιο σημαντικές υποκατηγορίες των συσκευών Internet of Things είναι οι συσκευές αυτοματισμού σπιτιών (**Home Automation Devices, Domotics [2]**). Οι συσκευές αυτές δίνουν στον χρήστη τους την δυνατότητα να διαχειριστεί διάφορες συσκευές του σπιτιού/γραφείου του. Οι συσκευές αυτές μπορεί να είναι συσκευές κλιματισμού, φωτισμός, συστήματα διασκέδασης (Home Theaters, Music Stereos, κτλ...), καθώς επίσης και συστήματα συναγερμού ή και διαχείρησης πρόσβασης. Το PiLock ανήκει στην τελευταία αυτή κατηγορία.

Συνήθως, οι συσκευές αυτές συνδέονται σε ένα κεντρικό κόμβο (Hub) προκειμένου να ελέγχονται όλες από ένα μοναδικό σημείο. Η δυνατότητα αυτή μπορεί να προστεθεί σε μία επόμενη έκδοση του PiLock (βλ. 14 Μελλοντική Επέκταση του PiLock). Την παρούσα χρονική στιγμή, δεν υπάρχει αυτή η δυνατότητα.

## 1.3 Σκοπός του PiLock

Το PiLock ανήκει στην κατηγορία συσκευών **”έξυπνου σπιτιού”** (Smart Home). Σκοπός του είναι να παρέχει στον χρήστη την δυνατότητα να ξεκλειδώνει εύκολα την εξώπορτα/πόρτα του σπιτιού/γραφείου του, μέσω του SmartPhone ή του SmartWatch του, όλα αυτά χρησιμοποιώντας το ασφαλέστερο δυνατόν περιβάλλον, προκειμένου να αποφευχθεί εισβολή τρίτων.

Μέσω του **PiLock Administration Control Panel (PiLock AdminCP)**, δίνεται στον διαχειριστή του συστήματος ένα εύχρηστο περιβάλλον διαχείρησης από το οποίο μπορεί εύκολα και γρήγορα να διαχειρίζεται το PiLock. Δίνεται δυνατότητα διαχείρησης των **εξουσιοδοτημένων χρηστών** (χρήστες που μπορούν να ξεκλειδώσουν την πόρτα μέσω του PiLock), δυνατότητα λήψης ζωτικής σημασίας πληροφοριών για το σύστημα, καθώς επίσης και της δυνατότητας ξεκλειδώματος της πόρτας απευθείας μέσω του πίνακα διαχείρησης, χωρίς να χρειάζεται να γίνει χρήση της εφαρμογής (AdminCP Unlock).

Ένας από τους στόχους, κατά τον σχεδιασμό του PiLock ήταν η διατήρηση του κόστους στο χαμηλότερο δυνατόν. Για να επιτευχθεί ο στόχος αυτός, χρησιμοποιήθηκε αυστηρά δωρεάν λογισμικό ανοικτού κώδικα, καθώς επίσης και εξαρτήματα εύκολα προσκομίσιμα (βλ. 2 Δομή του PiLock).

## 1.4 Έρευνα αγοράς - Καινοτομία του PiLock

Έπειτα από έρευνα που έγινε πάνω σε ήδη υπάρχοντες μηχανισμούς ξεκλειδώματος μέσω Raspberry Pi βρέθηκε οτι το PiLock είναι το πρώτο σύστημα ξεκλειδώματος που συνδέεται απευθείας πάνω στο κύκλωμα του θυροτηλεφώνου και χειρίζεται την κλειδαριά.

### 1.4.1 Μη Εμπορικές/DIY Εφαρμογές

Ανάμεσα στα συστήματα που βρέθηκαν, υπάρχει ένα σύστημα που συνδέεται απευθείας επάνω στην κλειδαριά της πόρτας, αλλά σε κλειδαριά διαφορετικού τύπου από ό,τι χρησιμοποιείται στις περισσότερες πολυκατοικίες στην Ελλάδα, δημιουργημένο από

έναν YouTuber γνωστό ως Hacker Shack<sup>1</sup>. Το συγκεκριμένο σύστημα χρησιμοποιείται σε κλειδαριές τύπου Deadbolt, αντί για κλειδαριές τύπου Electric Strike (βλ. 3 Συστήματα Ελέγχου Πρόσβασης Πολυκατοικιών/Σπιτιών).

Τηπάρχει ένα παρόμοιο, επίσης, σύστημα με την ονομασία "Pi-Lock", κατασκευασμένο από τον Paolo Bernasconi<sup>2</sup>, το οποίο χρησιμοποιεί Raspberry Pi αλλά προσφέρει λειτουργικότητα ξεκλειδώματος μέσω RFID, έναντι των ξεκλειδωμάτων μέσω Android, τα οποία προσφέρει το PiLock.

#### 1.4.2 Εμπορικές Εφαρμογές

Έπειτα από έρευνα που έγινε στις υπάρχουσες εμπορικές εφαρμογές έξυπνης κλειδαριάς, που μπορεί να προμηθευτεί ο οποιοισδήποτε, εξάγεται το συμπέρασμα ότι η πλειονότητα αυτών των εφαρμογών (παίρνοντας ως δείγμα το άρθρο με τις καλύτερες έξυπνες κλειδαριές του 2018, από το PC Magazine)<sup>[35]</sup> απαιτούν να ξοδευτεί αρκετά μεγάλο χρηματικό ποσό, σε σύγκριση με το ποσό που πρέπει να ξοδευτεί για να αγοραστούν τα εξαρτήματα του PiLock, και συνήθως απαιτούν αντικατάσταση της ήδη υπάρχουσας κλειδαριάς, το οποίο σημιαίνει ότι μπορεί να χρειαστούν παραπάνω χρήματα για την πρόσληψη τεχνικού που θα πραγματοποιήσει την αντικατάσταση.

#### 1.4.3 Διαφορές του PiLock με τις ήδη υπάρχουσες εφαρμογές

Το PiLock, εκτός του ότι είναι πολύ φυηνότερο σε σχέση με τις ήδη υπάρχουσες εμπορικές εφαρμογές που κυκλοφορούν, είναι πολύ ευκολότερο στην εγκατάσταση και μπορεί να εγκατασταθεί απευθείας στο ήδη υπάρχον σύστημα θυροτηλεφώνου που έχουν οι πολυκατοικίες, χωρίς να χρειαστεί να γίνει αλλαγή κλειδαριάς. Πέραν αυτού, με την προσθήκη συμβατότητας με Android Wear που έγινε στην έκδοση 0.3.0, είναι μία από τις πρώτες εφαρμογές παγκοσμίως που υποστηρίζουν ξεκλείδωμα πόρτας μέσω Smartwatch, και ίσως μία από τις εξελιγμένες, καθώς τα Project που υπάρχουν είναι σε πρωτογενή μορφή και δεν παρέχουν το πλήρες περιβάλλον διαχείρησης που παρέχει το PiLock.

---

<sup>1</sup><https://bit.ly/2LGTSJd>

<sup>2</sup><http://www.pi-lock.com/>

# Κεφάλαιο 2

## Δομή του PiLock

To PiLock χρησιμοποιεί Αρχιτεκτονική Πελάτη-Εξυπηρετητή (Client-Server Architecture).

### 2.1 Σύντομη Περιγραφή Λογισμικού Εξυπηρετητή - PiLock Server

Ο εξυπηρετητής αποτελείται από το Hardware που χρειάζεται προκειμένου να λειτουργήσει το PiLock, καθώς επίσης και το αντίστοιχο λογισμικό υπεύθυνο για την διαχείρηση του συστήματος ζεκλειδώματος. Πιο συγκεκριμένα, το λογισμικό είναι υπεύθυνο για:

- Την διαχείριση του Hardware υπεύθυνου για την λειτουργία του μηχανισμού ζεκλειδώματος.
- Την αυθεντικοποίηση των ήδη υπάρχοντων χρηστών.
- Την δημιουργία νέων χρηστών, ικανών για αυθεντικοποίηση (εξουσιοδοτημένοι χρήστες).
- Την τήρηση ιστορικού αυθεντικοποιήσεων (επιτυχών ή μή).

Το λογισμικό του εξυπηρετητή αναλύεται πλήρως στην ενότητα 8.

### 2.2 Σύντομη Περιγραφή Λογισμικού Πελάτη - PiLock Client

Η πλευρά του πελάτη αποτελείται από την εφαρμογή του PiLock, σχεδιασμένη για κινητά που τρέχουν Android, καθώς επίσης και από την εφαρμογή σχεδιασμένη για Android Wear Smartwatches.

Πιο συγκεκριμένα, οι εφαρμογές στο πεδίο του πελάτη είναι υπεύθυνες για:

- Σύνδεση στην πλατφόρμα του PiLock\*.
- Αποστολή αιτημάτων ζεκλειδώματος.
- Αποστολή αιτημάτων αλλαγής PIN\*.

Οι δυνατότητες που είναι σημειωμένες με τον αστερίσκο (\*) είναι διαθέσιμες αποκλειστικά στην εφαρμογή για κινητά (mobile app) και όχι στην εφαρμογή για Android Wear.

## 2.3 Υλικό - Hardware

Όπως αναφέρθηκε και στην εισαγωγή, ένας εκ των στόχων από τις πρώτες μέρες του σχεδιασμού του PiLock ήταν να υλοποιηθεί το Project με όσο το δυνατόν λιγότερο κόστος. Προκειμένου αυτό να είναι εφικτό, χρησιμοποιήσαμε υλικό εύκολα προσκομίσιμο και, όπου ήταν δυνατόν, Open Source Hardware.

### 2.3.1 Raspberry Pi Zero W

”Εγκέφαλος” όλης της κατασκευής είναι το Raspberry Pi Zero W (RPi Zero W), ένας υπολογιστής μοναδικής πλακέτας (Single Board). Σχεδιάζεται από το Raspberry Pi Foundation στην Αγγλία και η κυκλοφορία του ξεκίνησε τον Φεβρουάριο του 2017. Σκοπός του RPi Zero W είναι να συμπληρώσει το προηγούμενο μοντέλο, το Raspberry Pi Zero, φέρνοντας δυνατότητες συνδεσιμότητας WiFi 802.11n και BlueTooth 4.0 χωρίς Hardware κάποιου τρίτου (μέχρι προτίστως έπρεπε να χρησιμοποιηθεί κάποιο WiFi ή BlueTooth Dongle προκειμένου να υπάρξει αυτή η συνδεσιμότητα) [3].



Εικόνα 2.1: Το Raspberry Pi Zero W.

Στην ”καρδιά” του RPi Zero W υπάρχει ένας Broadcom BCM2835, 32-bit επεξεργαστής αρχιτεκτονικής ARMv6, χρονισμένος στο 1Ghz. Για μνήμη τυχαίας προσπέλασης χρησιμοποιούνται 512MB Low Power Double Data Rate 2 (LPDDR2) RAM. Πανω στο RPi Zero W δεν υπάρχει αποθηκευτικός χώρος, οπότε χρησιμοποιείται μια κάρτα MicroSD.

Ένα από τα σημαντικότερα σημεία ενός RPi Zero W είναι οι **Δέκτες Εισόδου/Εξόδου Γενικού Σκοπού** (General Purpose Input/Output, GPIO). Μέσω αυτών καθίσταται δυνατόν να συνδεθεί το RPi με μια πληθώρα εξωτερικών αισθητήρων, διακοπών (Relay Modules), πλακετών επέκτασης (γνωστά ως HATs), και εξαρτημάτων και να αντλήσει πληροφορίες από αυτά ή να τα ελέγξει.

### 2.3.2 Relay Module

Προκειμένου να μπορέσει να συνδεθεί το RPi με το ήδη υπάρχον σύστημα ξεκλειδώματος, χρειάζεται ένας ηλεκτρονικά ελεγχόμενος διακόπτης. Θα χρησιμοποιηθεί ένα **Relay Module**. Τα Relay Modules χρησιμοποιούνται ως διακόπτες προκειμένου να ελέγχονται κυκλώματα μέσω υπολογιστών/μικροελεγκτών, οι οποίοι λειτουργούν μέσω σημάτων μικρής ισχύος<sup>[5]</sup>.

Τα Relay Modules κυκλοφορούν σε πολλούς τύπους. Οι τρείς κυριότεροι είναι:

- 5V Compatible, Active Low.
- 5V/3.3V Compatible, Active High.
- 3.3V Compatible Active High/Low.

Το Raspberry Pi, εφόσον λειτουργεί σε λογική 3.3V, είναι συμβατό με τους 2 τελευτέους τύπους. Αν θελήσει ο χρήστης να χρησιμοποιήσει ένα Relay Module που να λειτουργεί σε λογική 5V και είναι Active Low, θα χρειαστεί να χρησιμοποιήσει ένα Arduino.

Τα Relay Modules αποτελούνται από ένα Relay τύπου SRD, έναν φωτοσυζευκτή (Optocoupler), ευθύνη του οποίου είναι να απομονώνει το κύκλωμα ώστε να μην επηρεάσει η υψηλή τάση (σε περίπτωση που χρησιμοποιείται από το σύστημα ξεκλειδώματος του κτηρίου) το υπόλοιπο κύκλωμα, ένα Transistor και μια δίοδο. Είναι σημαντικό να τονιστεί οτι καλό είναι να μην χρησιμοποιούνται Relay Modules που δεν φέρουν Φωτοσυζευκτή, καθώς μπορεί, σε περίπτωση που χρησιμοποιείται υψηλή τάση, να επηρεαστεί, ακόμα και να καεί, το Raspberry Pi ή/και το Arduino.

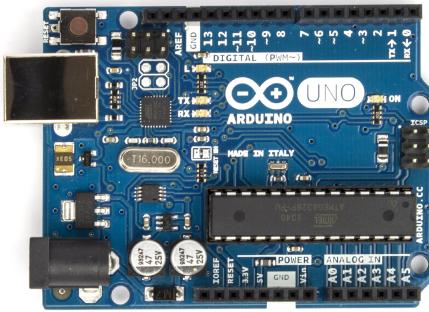
### 2.3.3 Arduino UNO

Το **Arduino UNO** είναι ένας Ανοικτού-Κώδικα (Open Source) μικροελεγκτής σχεδιασμένος από την Arduino.cc. Είναι βασισμένος πάνω στον ATmega328 microcontroller της Atmel. Μπορεί να χρησιμοποιηθεί προκειμένου να χειρίζεται και να αντλεί πληροφορίες από διάφορα εξαρτήματα στον φυσικό κόσμο. Εξαιτίας της μεγάλης ευελιξίας του έχει γίνει μία από τις δημοφιλέστερες επιλογές για κατασκευαστές, οι οποίοι το χρησιμοποιούν για μια τεράστια γκάμια εφαρμογών<sup>[6]</sup>.

Το Arduino UNO μπορεί να χρησιμοποιηθεί σε περίπτωση που δεν χρησιμοποιηθεί κάποιο Relay συμβατό με το Raspberry Pi (βλ. 2.3.2 Relay Module), αρκεί να λειτουργεί με λογική 5V.

Μπορεί, έναντι του Arduino UNO, και προκειμένου να εξοικονομηθεί χώρος, να χρησιμοποιηθεί ένα Arduino Nano, το οποίο έχει όλες τις αναγκαίες λειτουργίες για την λειτουργία του PiLock.

Ρεύμα για την λειτουργία του Arduino παρέχεται από την θύρα Micro USB του RPi, και μέσω αυτού δίνεται ρεύμα και σε οποιοδήποτε Relay Module συνδεθεί με αυτό. Για να γίνει αποστολή δεδομένων από το RPi στο Arduino χρησιμοποιείται η Σειριακή Θύρα (Serial Port) του Arduino.



Εικόνα 2.2: Arduino Uno Rev3, oomlout (2015), Flickr, CC BY-SA 2.0

### 2.3.4 Λοιπό Hardware

Προκειμένου να συναρμολογηθεί η κατασκευή ωστε χρειαστουν κάποια συγκεκριμένα υλικά, εύκολα προμηθεύσιμα από διάφορα μαγαζιά πώλησης υλικών για ηλεκτρονικές κατασκευές.

#### Κουτί Κατασκευής (Project Box):

Ανάλογα τον τρόπο σύνδεσης που θα χρησιμοποιηθεί για την σύνδεση του RPi με το Relay Module, και ανάλογα με το αν είναι συμβατό το Relay Module με λογική 3.3V, θα χρειαστεί διαφορετικό μέγεθος κουτιού κατασκευής.

**Σύνδεση χωρίς χρήση Arduino:** Ο προεπιλεγμένος τρόπος σύνδεσης, από την έκδοση 0.3.1 και μετά είναι χωρίς την χρήση Arduino. Έπειτα από μετρήσεις βρέθηκε οτι το κατάλληλο κουτί κατασκευής έχει διαστάσεις 10cm x 10cm.

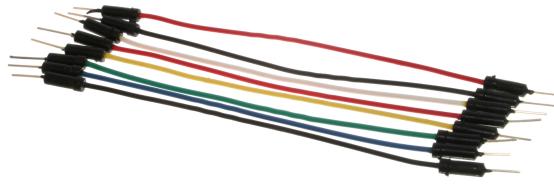
**Σύνδεση με Arduino:** Εφόσον χρειάζεται να γίνει σύνδεση με Arduino (προκειμένου να μπορεί να λειτουργήσει το Relay Module), έπειτα από μετρήσεις βρέθηκε οτι το κατάλληλο κουτί κατασκευής έχει διαστάσεις 18cm x 14cm.

#### Καλώδια σύνδεσης:

Για να συνδεθεί το Relay Module με το RPi (ή το Arduino), θα χρειαστούν κάποια συγκεκριμένα καλώδια σύνδεσης γνωστά ως Jumper Wires. Τα Jumper Wires κάνουν εύκολη την σύνδεση σε διάφορα εξαρτήματα καθώς δεν χρειάζονται συγκόλληση [7].

**Σύνδεση χωρίς χρήση Arduino:** Θα χρειαστούν τουλάχιστον 3 Jumper Wires Female-Male (ή Female-Female, σε περίπτωση χρήσης του Male Header).

**Σύνδεση με Arduino:** Θα χρειαστούν τουλάχιστον 3 Jumper Wires Female-Male, αν χρησιμοποιηθεί Arduino UNO ή 3 τουλάχιστον καλώδια Female-Female, αν χρησιμοποιηθεί Arduino Nano. Επίσης, θα χρειαστεί ένα καλώδιο Micro USB-B to USB-A (OTG Cable) και ένα καλώδιο USB-A to USB-B αν χρησιμοποιηθεί ένα Arduino UNO ή ένα καλώδιο USB-A to Micro USB-B σε περίπτωση χρήσης Arduino Nano.



Εικόνα 2.3: Jumper Wires (Male-Male), oomlout (2009), Flickr, CC BY-SA 2.0

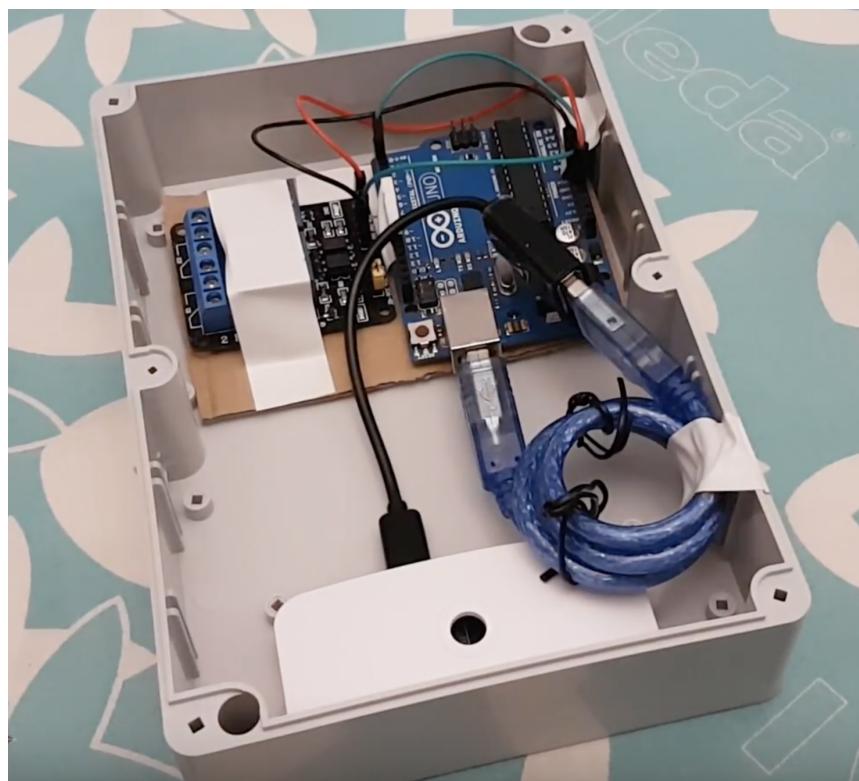
## 2.4 Κόστος Κατασκευής του PiLock

Με βάση την παραπάνω λίστα υλικών που θα χρησιμοποιηθούν για την συναρμολόγηση του PiLock, μπορεί να εξαχθεί ένα τελικό κόστος για την προμήθεια των υλικών αυτών. Να σημειωθεί οτι η εξαγωγή των τιμών αυτών έγινε έπειτα από έρευνα σε διάφορα ηλεκτρονικά μαγαζιά, είτε Ελληνικά, είτε του εξωτερικού, προκειμένου να χρατηθεί σε όσο το δυνατόν χαμηλότερα πλαίσια το τελικό κόστος.

Τλικό	Κόστος
Raspberry Pi Zero W*	17.46€
Project Box (18x14)	4.90€
Relay Module	2.75€
Arduino Uno (Κλώνος)	5.00€
Καλώδια	0.70€
Τελικό Κόστος:	30.81€

\*: Συμπεριλαμβάνεται Καλώδιο OTG, GPIO Headers

Να σημειωθεί οτι, όπως είναι αντιληπτό, στο παραπάνω κόστος συμπεριλαμβάνεται και το Arduino. Εάν αγοραστεί συμβατό με το Raspberry Pi, Relay Module, το κόστος κατεβαίνει στα 23,81€, καθώς δεν απαιτείται η χρήση Arduino και, κατ'επέκταση, απαιτείται και μικρότερο Project Box.



Εικόνα 2.4: Το PiLock, χρησιμοποιούντας Arduino. Στο συγκεκριμένο παράδειγμα υπάρχει στα αριστερά το Relay Module, στα δεξιά ένα Arduino Uno, και από κάτω, το Raspberry Pi.

## Κεφάλαιο 3

# Συστήματα Ελέγχου Πρόσβασης Πολυκατοικιών/Σπιτιών

Πριν να εξηγηθεί ο τρόπος κατασκευής και λειτουργίας του PiLock, είναι αναγκαίο να αναφερθεί ο τρόπος λειτουργίας των περισσότερων κλειδαριών σπιτιών, πολυκατοικιών ή και γραφείων.

Το σύστημα ξεκλειδώματος που χρησιμοποιείται στις περισσότερες κατοικίες στην Ελλάδα αποτελείται από 2 εντελός ξεχωριστά και ανεξάρτητα συστήματα: Το σύστημα του ψυροτηλεφώνου, δηλαδή το σύστημα μέσω του οποίου γίνεται η αναγνώριση του επισκέπτη (μέσω φωνής ή/και εικόνας), και το σύστημα ενεργοποίησης της κλειδαριάς. Στην παρούσα διατριβή θα αναφερθούμε αποκλειστικά στο δεύτερο σύστημα.

Οι ηλεκτρικές κλειδαριές που χρησιμοποιούνται σε πολυκατοικίες συνήθως αποτελούνται από ένα μάνταλο το οποίο, όταν το σύστημα ενεργοποιηθεί μέσω ρεύματος, απελευθερώνεται με αποτέλεσμα να μπορεί ελεύθερα η πόρτα να ανοίξει.



Εικόνα 3.1: Ηλεκτρική κλειδαριά πολυκατοικίας με σύστημα καταγραφής κατάστασης κλειδώματος

Η ενεργοποίηση του παραπάνω συστήματος γίνεται μέσω ενός διακόπτη αναρτημένου πάνω στο ψυροτηλέφωνο του κάθε διαμερίσματος. Η τροφοδοσία των συστημάτων αυτών μπορεί να γίνεται είτε μέσω απευθείας τροφοδοσίας από το ηλεκτρικό δίκτυο, είτε μέσω κάποιου μετασχηματιστή, σε χαμηλότερες τάσεις.

Προκειόντων να καταστεί δυνατόν να ελεγχθεί το σύστημα ξεκλειδώματος, θα πρέπει να αναρτηθεί ένας δεύτερος διακόπτης, ο οποίος να λειτουργεί παράλληλα με

τον πρώτο.

Για να γίνει αυτό, θα πρέπει να εντοπιστεί το κύκλωμα που συνδέεται με τον ήδη υπάρχοντα διακόπτη ξεκλειδώματος (του θυροτηλεφώνου) και να τοποθετηθούν 2 καλώδια σε παράλληλη σύνδεση με τον ήδη υπάρχοντα διακόπτη.

Για τον εντοπισμό θα πρέπει να αποσυνδεθεί ο διακόπτης από τον τοίχο, λύνοντας τις βίδες οι οποίες τον συγχρατούν. Προκειμένου να γίνει υπο ασφαλές συνθήκες, συνίσταται να απενεργοποιηθεί η παροχή ρεύματος στο παρόν τμήμα της οικίας, μέχρι να ολοκληρωθεί η εγκατάσταση. Αυτό μπορεί να γίνει είτε κατεβάζοντας την ασφάλεια που αντοιστοιχεί στο συγκεκριμένο τμήμα της κατοικίας, από τον ηλεκτρικό πίνακα, είτε απενεργοποιώντας τον γενικό διακόπτη τροφοδοσίας της κατοικίας.



Εικόνα 3.2: Παράδειγμα εγκατάσταση διακόπτη σε παράλληλη σύνδεση. Στην συγκεκριμένη εγκατάσταση διαχρίνεται το κουδούνι-μετασχηματιστής (Κουτί στον τοίχο) που χρησιμοποιείται για το σύστημα του θυροτηλεφώνου, και διαχρίνονται και 2 καλώδια που έχουν συνδεθεί παράλληλα με τον διακόπτη που χρησιμοποιείται για το κουδούνι.

Η τελική εγκατάσταση θα υπογραμμιστεί σε επόμενο κεφάλαιο.



Εικόνα 3.3: Το PiLock, όταν έχει εγκαταστηθεί. Τα καλώδια που αναρτήθηκαν στο προηγούμενο βήμα έχουν συνδεθεί πάνω στο Relay Module του PiLock, σε αυτό το παράδειγμα, χωρίς την χρήση Arduino.

# Κεφάλαιο 4

## Νέος μηχανισμός ξεκλειδώματος

Στο προηγούμενο κεφάλαιο υπογραμμίστηκαν τα συστατικά ενός συστήματος ξεκλειδώματος πόρτας πολυκατοικίας/σπιτιού. Στο παρόν κεφάλαιο θα αναλυθεί ο τρόπος σύνδεσης των εξαρτημάτων (βλ 2.3 Υλικό - Hardware), και ο προγραμματισμός τους ώστε να μπορεί να πυροδοτηθεί ξεκλειδωματικά της πόρτας μέσω υπολογιστή.

### 4.1 Προετοιμασία του Raspberry Pi

Πρώτο βήμα πριν να γίνει η οποιαδήποτε σύνδεση μεταξύ εξαρτημάτων είναι να γίνει η εγκατάσταση της τελευταίας έκδοσης του Raspbian Lite στο Raspberry Pi Zero W, καθώς επίσης και να συνδεθεί με το internet. Χρησιμοποιείται η έκδοση Lite έναντι της πλήρης έκδοσης, καθώς δεν χρειάζεται γραφικό περιβάλλον όπως επίσης και τα περισσότερα που υπάρχουν προεγκατεστημένα στην πλήρη έκδοση.

Αφότου γίνει η εγκατάσταση του λειτουργικού συστήματος στην κάρτα MicroSD, μπορεί να γίνει η σύνδεση στο Internet είτε Headlessly (χωρίς, δηλαδή, να χρειαστεί να συνδεθεί οινόνη στο RPi), βάζοντας το αρχείο στο `boot` partition (διαμέρισμα) της κάρτας μνήμης, και εφαρμόζοντας το παρακάτω configuration, μέσα στο αρχείο `wpa_supplicant.conf`<sup>[8]</sup>:

```
1 network={  
2     ssid="YOUR_NETWORK_NAME"  
3     psk="YOUR_PASSWORD"  
4 }
```

Στο πεδίο `ssid` πρέπει να μπει το όνομα του δικτύου στο οποίο πρόκειται να συνδεθεί το RPi. Στο πεδίο PSK πρέπει να γίνει τοποθέτηση του κλειδιού πρόσβασης του δικτύου. Σε περίπτωση που δεν χρησιμοποιείται κρυπτογραφία στο δίκτυο (δεν προτείνεται καθώς μπορεί να συμβεί υποκλοπή δεδομένων από τρίτους), μπορεί να εισαχθεί το παρακάτω configuration στο ίδιο αρχείο:

```
1 network={  
2     ssid="YOUR_NETWORK_NAME"  
3     key_mgmt=NONE  
4 }
```

Τέλος, πρέπει να γίνει ενεργοποίηση του SSH Daemon στο Raspbian, προκειμένου να είναι εφικτή η απομιαρυσμένη σύνδεση στο RPi, μέσω τοπικού δικτύου. Λόγω

κατασκευής δικτύων bot (botnets) από διάφορους κακόβουλους χρήστες προκειμένου να γίνουν επιθέσεις DDOS (Distributed Denial of Service) από συσκευές Internet of Things που χρησιμοποιούν τα προεπιλεγμένα (default) στοιχεία πρόσβασης (Username, Password), προς διάφορους στόχους, από τον Νοέμβριο του 2016 είναι απενεργοποιημένος εξ' αρχής ο SSH Daemon και πρέπει να ενεργοποιηθεί από τον χρήστη, αν τον χρειάζεται [9]. Για να γίνει αυτό, πρέπει να δημιουργηθεί ένα άδειο αρχείο με το όνομα `ssh` μέσα στο boot partition της κάρτας μνήμης του RPi.

Αφότου γίνει η πρώτη εκκίνηση του RPi και βεβαιωθεί οτι υπάρχει ενεργή σύνδεση στο διαδίκτυο, πρέπει να γίνει σύνδεση στο Raspberry Pi μέσω SSH (Username: pi, Password: raspberry) [10].

Προκειμένου να βρεθεί η διεύθυνση IP που χρησιμοποιεί το RPi, ο χρήστης μπορεί να συμβουλευτεί τον πίνακα DHCP του οικιακού Router του, είτε να χρησιμοποιήσει μια εφαρμογή διαγνωστικών δικτύου από κάποιο Smartphone ή H/Y (για Android, και iOS, προτείνεται η εφαρμογή Fing) [13].

Έπειτα, και αφότου γίνει γνωστή η διεύθυνση IP του Raspberry Pi, πρέπει να γίνουν τα ακόλουθα βήματα, προκειμένου να ενημερωθεί πλήρως το Raspbian:

#### 4.1.1 Αλλαγή των προεπιλεγμένων στοιχείων πρόσβασης

Όπως αναφέρθηκε προηγουμένως, προκειμένου να μην υπάρξει στο μέλλον κίνδυνος επίθεσης, πρέπει να γίνει αλλαγή των προεπιλεγμένων στοιχείων πρόσβασης στο Raspbian. Πρέπει να εκτελεστεί η εντολή `passwd`, και να γίνει εισαγωγή ενός νέου κωδικού πρόσβασης. Ο νέος κωδικός, προκειμένου να είναι ασφαλής, πρέπει να αποτελείται από τουλάχιστον 12 χαρακτήρες, να μην περιέχει μέσα ονόματα, ονόματα από μέρη, ή γενικά λέξεις οι οποίες υπάρχουν μέσα σε λεξικά, και τέλος όταν πρέπει να περιέχει πεζά γράμματα, κεφαλαία, αριθμούς, και σύμβολα [11].

#### 4.1.2 Ενημέρωση του Raspbian

Προκειμένου να εξασφαλιστεί η βέλτιστη λειτουργία και η μέγιστη ασφάλεια στο σύστημα, χρειάζεται να γίνει ενημέρωση των πακέτων του λειτουργικού συστήματος. Πρέπει να γίνει εκτέλεση των επόμενων 2 εντολών [12]:

- 1        `sudo apt-get update`
- 2        `sudo apt-get dist-upgrade`

#### 4.1.3 Ανάθεση στατικής διεύθυνσης IP

Από προεπιλογή, το Raspberry Pi λαμβάνει μια IP διεύθυνση μέσω ενός DHCP εξυπηρετητή στο τοπικό δίκτυο. Αυτή η διεύθυνση IP μπορεί να αλλάξει, μόλις λήξει ο χρόνος μίσθωσης της (DHCP Lease Time). Προκειμένου να λειτουργήσει το PiLock, όταν χρειαστεί η διεύθυνση IP να γίνει στατική, έτσι ώστε να μην αλλάζει.

Αφότου βρεθεί μια διαθέσιμη διεύθυνση IP στο τοπικό δίκτυο (που να είναι εκτός του εύρους διευθύνσεων που να μπορεί να αναθέτει ο DHCP Server), και αφού καταγραφεί και η μάσκα υποδικτύου που αντιστοιχεί στο συγκεκριμένο υποδίκτυο, καθώς επίσης και η διεύθυνση προεπιλεγμένης πύλης, πρέπει να γίνει ανάθεση της συγκεκριμένης διεύθυνσης IP μέσω του DHCP Client Daemon (`dhcpcd`). Για να

γίνει αυτό πρέπει να γίνει επεξεργασία του αρχείου `/etc/dhcpcd.conf` και να γίνει πρόσθεση των παρακάτω γραμμών, στο τέλος του αρχείου:

```
1   interface wlan0
2     static ip_address=192.168.1.2/24
3     static routers=192.168.1.1
4     static domain_name_servers=192.168.1.5
5     static domain_search=home.lan
6     noipv6
```

Στο παραπάνω παράδειγμα, ανατίθεται η διεύθυνση 192.168.1.2, με μάσκα υποδικτύου την 255.255.255.0 (24 bits μάσκας). Ορίζεται επίσης ως προεπιλεγμένη πύλη (Router) η συσκευή με την διεύθυνση 192.168.1.1 και ως διακομιστής DNS η συσκευή 192.168.1.5. Προαιρετικά, μπορεί να οριστεί και το Search Domain. Σε αυτό το παράδειγμα το ορίζεται ως Search Domain το home.lan. Με την τελευταία γραμμή, υποχρεώνεται το DHCPCD να απενεργοποιηθεί, για διευθύνσεις IPv6.

Αφότου γίνει η αλλαγή των προεπιλεγμένων στοιχείων πρόσβασης και η ενημέρωση του συστήματος, πρέπει να απενεργοποιηθεί το σύστημα προκειμένου να συνδεθεί με το Relay.

## 4.2 Σύνδεση με το Relay

Για να γίνει η σύνδεση του Raspberry Pi με το Relay, πρέπει να επιλεγεί ένας από τους 2 τρόπους σύνδεσης, ανάλογα με τη τι Relay Module θα χρησιμοποιηθεί (βλ. 2.3.2 Relay Module).

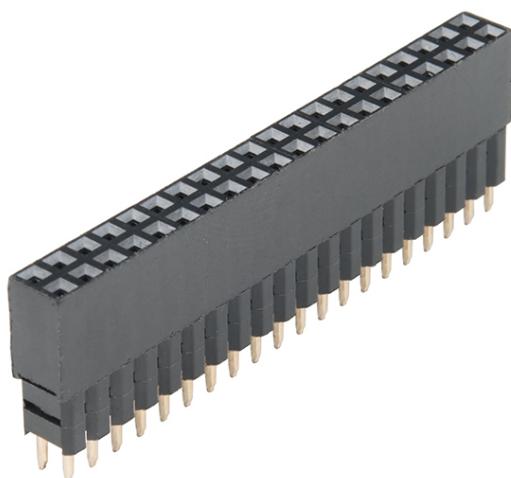
### 4.2.1 Σύνδεση χωρίς την χρήση Arduino

Προκειμένου να γίνει σύνδεση του RPi με το Relay Module, θα χρειαστεί να κολληθούν κεφαλές υποδοχής για Jumper Wires στα GPIO του Raspberry Pi. Μπορούν να χρησιμοποιηθούν είτε Female, είτε Male τύπου Headers, αλλά από αυτό θα εξαρτηθεί τι Jumper Wires θα χρειαστούν για να συνδεθεί το RPi με το Relay Module (Male-Female αν χρησιμοποιηθεί Female Header, Female-Female αν χρησιμοποιηθεί Male Header).

Αφότου γίνει η κόλληση των Headers, μπορούν να χρησιμοποιηθούν τα αντίστοιχα καλώδια προκειμένου να συνδεθεί το Relay Module. Πρέπει ο χρήστης που θα το συνδέσει να συμβουλευτεί το διάγραμμα των GPIO Pins (γνωστό ως Pinout). Στο συγκεκριμένο παράδειγμα (Εικόνα 4.2), έχει συνδεθεί στο GPIO Pin 18 (πράσινο καλώδιο). Για παροχή ρεύματος χρησιμοποιείται το 3V3 Pin (κόκκινο καλώδιο) και για Ground χρησιμοποιείται ένα από όλα τα GND Pins (μαύρο καλώδιο).

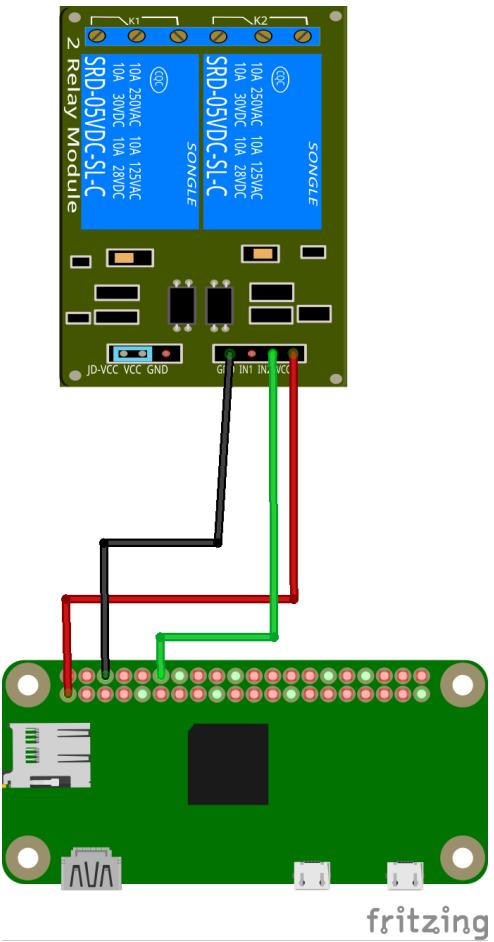
### 4.2.2 Σύνδεση με την χρήση Arduino

Αφότου γίνει Upload το Arduino Script που χρειάζεται για την λειτουργία του PiLock (βλ. 7.2 Σενάριο Ξεκλειδώματος Arduino), μπορεί να γίνει σύνδεση του Arduino με το Relay. Το Relay Board μπορεί να συνδεθεί σε ένα από όλα τα Digital Pins του Arduino (πράσινο καλώδιο) (Εικόνα 4.3). Ρεύμα δίνεται μέσω του 5V Power Pin του Arduino και το Ground συνδέεται σε ένα εκ των τριών GND Pins του Arduino.

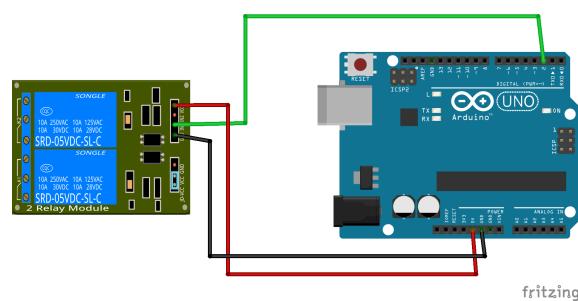


Εικόνα 4.1: Female GPIO Headers, SparkFun Electronics (2016), Flickr, CC-BY 2.0

Αφότου συνδεθεί το Relay Module με το Arduino, πρέπει να συνδεθεί το Arduino με το Raspberry Pi. Αυτό γίνεται συνδέοντας το καλώδιο OTG (βλ. 2.3.4 Σύνδεση με Arduino:) με το RPi, και την άλλη άκρη του με το Καλώδιο USB του Arduino.



Εικόνα 4.2: Σύνδεση ενός Raspberry Pi Zero W με ένα Relay Module



Εικόνα 4.3: Σύνδεση ενός Arduino με ένα Relay Module.

# Κεφάλαιο 5

## Προγραμματιστικό Περιβάλλον - Τεχνολογίες που Χρησιμοποιήθηκαν

Στο παρόν κεφάλαιο θα αναφερθούν και θα αναλυθούν τα εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη του PiLock, καθώς επίσης και στην διαχείρηση του έργου ανάπτυξης.

### 5.1 Η σημασία χρήσης δωρεάν λογισμικού ανοικτού κώδικα κατά την ανάπτυξη του PiLock

Ως ”Λογισμικό Ανοικτού Κώδικα” (Open Source Software) ορίζεται το λογισμικό του οποίου ο πηγαίος κώδικας είναι διαθέσιμος ελεύθερα προς το κοινό προκειμένου να μπορεί να τροποποιηθεί, να αναβαθμιστεί ή να μελετηθεί. Ο πηγαίος κώδικας, για τον απλό χρήστη είναι ένα τμήμα του λογισμικού που δεν έχει δει ποτέ. Σε έργα ανοικτού κώδικα, μπορεί ο οποιοσδήποτε να προτείνει διορθώσεις, αναβαθμίσεις ή προσθήκη χαρακτηριστικών<sup>[14]</sup>.

Εξαιτίας αυτού του χαρακτηριστικού, και των αδειών που το υποστηρίζουν, το λογισμικό ανοικτού κώδικα μπορεί να χρησιμοποιηθεί για οποιοδήποτε σκοπό επιθυμεί ο χρήστης, χωρίς να περιορίζεται από κάποια άδεια χρήσης. Επίσης, παρέχει αυξημένη ασφάλεια, εφόσον μπορεί να δοκιμαστεί και να μελετηθεί από τους προγραμματιστές ο πηγαίος κώδικας του. Στο λογισμικό κλειστού κώδικα, είναι αδύνατον να μελετηθεί και να τροποποιηθεί από τρίτους ο κώδικας του, και κατ’ επέκταση, εφόσον υπάρξει ένα κενό ασφαλείας θα πάρει συνήθως αρκετά περισσότερο χρόνο μέχρι να κυκλοφορήσει μια ενημέρωση ασφάλειας. Τέλος, εξαιτίας της ευελιξίας που παρέχει, το λογισμικό ανοικτού κώδικα μπορεί να τροποποιηθεί προκειμένου να μπορέσει καλύτερα να καλύψει τις ανάγκες του χρήστη<sup>[14], [15]</sup>.

Μία υποκατηγορία του λογισμικού ανοικτού κώδικα είναι το **Δωρεάν Λογισμικό Ανοικτού Κώδικα** (Free and Open Source Software, FOSS), το οποίο επιτρέπει στον χρήστη να το κατεβάσει και να το χρησιμοποιήσει χωρίς κάποιο κόστος. Το λογισμικό που χρησιμοποιήθηκε για την ανάπτυξη του PiLock ανήκει στην κατηγορία αυτή.

## 5.2 Άδεια Χρήσης του PiLock

Το PiLock είναι ένα έργο ανοικτού κώδικα και όλα τα πακέτα λογισμικού που απαιτούνται για την δημιουργία του ανήκουν, και αυτά, σε αυτή την κατηγορία. Η άδεια που επιλέχθηκε να υπόκειται το PiLock είναι η GNU General Public Licence, έκδοση 3 (GNU GPLv3)<sup>1</sup>.

Η συγκεκριμένη άδεια επιτρέπει στον οποιονδήποτε να χρησιμοποιήσει το λογισμικό, εμπορικά ή μή, και για οποιονδήποτε σκοπό. Επίσης επιτρέπει στον οποιονδήποτε να το μελετήσει ή/και να το τροποποιήσει, να το βελτιώσει προκειμένου να επωφεληθεί η κοινότητα, και να διανείμει αντίγραφά του. Προϋπόθεση της GPL είναι οποιοδήποτε αντίγραφο, παράγογο ή/και προϊόν ενός προγράμματος που διανείμεται υπό αυτήν, να κυκλοφορεί και αυτό υπό την ίδια άδεια και να αναφέρονται οι αλλαγές<sup>[40]</sup>.

## 5.3 Διαχείρηση του έργου

### 5.3.1 Version Control

Καθ' όλη την διαδικασία ανάπτυξης του PiLock χρησιμοποιήθηκε σύστημα Version Control. Τα Συστήματα Version Control βοηθούν τον/τους προγραμματιστή/ές καθώς προσδίδουν ασφάλεια και ευελιξία κατά την ανάπτυξη και την συντήρηση ενός έργου. Το Git είναι το λογισμικό Version Control που χρησιμοποιήθηκε κατά την ανάπτυξη του PiLock.

Το Git δημιουργήθηκε από τον Linus Torvalds το 2005 προκειμένου να τον βοηθήσει στην ανάπτυξη του Linux Kernel<sup>[17]</sup>. Διάφοροι άλλοι συνεισφέροντες προς το Linux Kernel βοήθησαν στην ανάπτυξη του Git, κατά την πρώτη περίοδο της ανάπτυξής του. Αποτελεί δωρεάν λογισμικό ανοικτού κώδικα και διανείμεται υπό την άδεια GNU General Public Licence, έκδοση 2<sup>[16]</sup>.

Πιο συγκεκριμένα, εξαιτίας της ικανότητάς του Git να διατηρεί ιστορικό για όλα τα αρχεία ενός έργου, το έργο μπορεί να επανέλθει σε μία προηγούμενη κατάστασή του, ανά πάσα στιγμή. Οι "καταστάσεις" είναι γνωστές ως "Commits". Με αυτό τον τρόπο, δεν απορρίπτεται κώδικας, πολλές φορές πολύτιμος για την ανάπτυξη ενός έργου. Μέσω των commits, μπορεί κάποιος να βεβαιωθεί ποιος έκανε αλλαγές στον κώδικα σε οποιοδήποτε σημείο επιθυμεί. Αξίζει να αναφερθεί η δυνατότητα της ψηφιακής υπογραφής των commits μέσω του GNU Privacy Guard (GPG), προκειμένου να αποφευχθεί προσωποποίηση<sup>[18]</sup>. Επίσης, μέσω του συστήματος Staging του Git, ο προγραμματιστής γνωρίζει τι ακριβώς κρατείται σε ένα νέο Commit, όποτε καταχωρηθεί. Με το Branching System του Git, καθίσταται δυνατόν να τροποποιείται ή να προστίθεται κώδικας και νέα features χωρίς να τροποποιείται ο Stable κώδικας του έργου (για παράδειγμα, το κάθε release χρησιμοποιεί υποχρεωτικά νέο branch), ο οποίος "ενημερώνεται" στο τέλος του κάθε release/feature, προκειμένου να αποφευχθούν προβλήματα. Τέλος, το Git διευκολύνει σημαντικά την συνεργασία μεταξύ προγραμματιστών καθώς μπορεί ο κάθε προγραμματιστής να δουλεύει το δικό του "κοιμάτι" κώδικα, χωρίς να επηρεάζει την πρόοδο των υπολοίπων προγραμματιστών που δουλεύουν πάνω στο έργο.

Στην ανάπτυξη του PiLock, το λογισμικό της πλευράς του Εξυπηρετητή, του Πελάτη καθώς επίσης και τα σενάρια ξεκλειδώματος (Unlock Scripts) διαχειρίζονται ξεχωριστά σε διαφορετικά αποθετήρια. Συγκεκριμένα, τα σενάρια ξεκλειδώματος

<sup>1</sup><https://www.gnu.org/licenses/gpl-3.0.en.html>

αποτελούν Submodule του λογισμικού του εξυπηρετητή, δηλαδή μπορεί να εμφαλευθεί ως ξεχωριστό αποθετήριο μέσα σε ένα ήδη υπάρχον αποθετήριο, ως κομμάτι του.

Προκειμένου να γίνει σωστή διαχείρηση της διαδικασίας ανάπτυξης, ακολουθήσης κάποιοι κανόνες. Οι κανόνες αυτοί διασφαλίζουν την ακαρεότητα του κώδικα ανα πάσα στιγμή κατά την ανάπτυξη. Πιο συγκεκριμένα:

- Η σταθερή (Stable) έκδοση του κώδικα βρίσκεται στο master branch.
- Όποια αλλαγή πρόκειται να γίνει στον κώδικα, είτε αυτή είναι hotfix, είτε κάποιο νέο feature, είτε documentation, θα πρέπει να γίνεται αποκλειστικά σε νέο branch με χαρακτηριστικό τίτλο, ο οποίος να ξεκινά από το ανάλογο πρόθεμα (prefix). Συγκεκριμένα, αν πρόκειται για νέο feature να χρησιμοποιείται το "feature" prefix, αν πρόκειται για bugfix να χρησιμοποιείται το "bug" ή το "hotfix" prefix, και αν πρόκειται για αλλαγή στο documentation ή στο Readme, να χρησιμοποιείται το "doc" prefix. (Πχ. feature/pin\_changing)
- Τα νέα Branches, εφόσον ελεγχθούν, θα πρέπει να συγχωνεύονται στο αντίστοιχο branch στο οποίο απευθύνονται (βλ. παρακάτω).
- Νέα λειτουργικότητα (νέα features) προστίθεται μόνο στα νέα releases (στο branch του εκάστοτε νέου release). Τα hotfix κανώς επίσης και διάφορα bugs μπορούν να συγχωνεύονται απευθείας με το master, αρκεί να έχουν ελεγχθεί εξουνχιστικά πρώτα και να είναι υψηλής προτεραιότητας.
- Το master branch, κανώς επίσης και τα branches για νέα releases θα πρέπει να είναι κλειδωμένα και να δέχονται συγχονεύσεις μόνο εφόσον περάσει από έγκριση ο κώδικας μέσω κάποιου Pull Request ή Merge Request.

Αρχικά, μέχρι την πρώτη δημόσια έκδοσή του (0.2.0), ο κώδικας του PiLock φιλοξενούνταν στον προσωπικό εξυπηρετητή του δημιουργού, και τα αποθετήρια του διαχειρίζονταν μέσω του δωρεάν λογισμικού διαχείρησης αποθετηρίων Git γνωστό ως GitLab (<https://about.gitlab.com>). Αργότερα, από την πρώτη δημόσια έκδοσή του PiLock και μετά, ξεκίνησε να χρησιμοποιείται το GitHub (<https://github.com>) ως χώρος φιλοξενίας του έργου και των αποθετηρίων του.

### 5.3.2 Issue Tracking

Κατά την ανάπτυξη του PiLock, έγινε χρήση τόσο του συστήματος Issue Tracking του GitLab, όσο και του GitHub. Τα συστήματα Issue Tracking, χρησιμοποιούνται για να κρατάνε μια λίστα με διάφορα "Ζητήματα" που προκύπτουν κατά την ανάπτυξη ενός έργου ή που προέρχονται από εξωτερικούς χρήστες. Στην 2η κατηγορία ανήκουν διάφορα αιτήματα νέας λειτουργικότητας, ή διάφορα bugs τα οποία μπορεί να έχουν αναφερθεί, κατά την χρήση του λογισμικού ή κατά την διάρκεια δοκιμών (testing). Issues επίσης μπορεί να προκύψουν από εξωτερικούς χρήστες ως απλά ερωτήματα για την χρήση του λογισμικού.

Τα Issues, έχουν 2 κύριες καταστάσεις: Open (Ανοικτό), Closed (Κλειστό/Ολοκληρωμένο). Τα ανοικτά issues είναι τα αυτά που ακόμια δεν έχουν ικανοποιηθεί οι απαιτήσεις τους, ή είναι σε διαδικασία ανάπτυξης. Τα κλειστά issues είναι τα εκπληρωμένα issues ή όσα issues δεν είναι δυνατόν να εκπληρωθούν για κάποιο λόγο, και δεν πρόκειται να αναπτυχθούν άλλο.

Προκειμένου να μπορούν να κατηγοριοποιηθούν τα Issues ενός έργου, και να τα αναλάβουν, πολλές φορές διαφορετικές ομάδες, χρησιμοποιούνται οι ετικέτες (Labels). Κάποια χαρακτηριστικά παραδείγματα χρήσης ετικετών είναι για να χωριστούν τα issues που απαιτούν νέα λειτουργικότητα από τα issues που χρησιμοποιούνται προκειμένου να επιδιορθωθεί ένα bug.

Όπως αναφέραμε στην αρχή, κατά την ανάπτυξη του PiLock, χρησιμοποιήθηκε Issue Tracking προκειμένου να οργανωθεί περισσότερο η διαδικασία της ανάπτυξης. Αξίζει να τονιστεί οτι έπειτα από την πρώτη δημόσια έκδοση του PiLock, μπορεί ο οποιοσδήποτε χρήστης του να συνεισφέρει στην συνεισφορά νέας λειτουργικότητας ή στην αναφορά και επίλυση bugs που υπάρχουν στο σύστημα.

Το κάθε issue καταχωρείται στο αντίστοιχο milestone. Ως ”Milestone” (Ορόσημο) ορίζεται ένα σημαντικό σημείο κατά την ανάπτυξη του έργου. Τα Milestones δημιουργούνται από τους συντηρητές ή τους διαχειριστές ενός έργου και χρησιμοποιούνται προκειμένου να οργανωθεί καλύτερα η ανάπτυξη του έργου και για να κατηγοριοποιηθούν τα issues. Για παράδειγμα, ως milestones συνήθως ορίζονται οι νέες εκδόσεις ενός έργου, πριν να γίνουν stable. Αφότου γίνουν Stable, το milestone κλείνει.

## 5.4 Προγραμματιστικό Περιβάλλον

### 5.4.1 Γλώσσες Προγραμματισμού/Markup

Το λογισμικό που χειρίζεται το Business Logic του εξυπηρετητή του PiLock ήταν, αρχικά, γραμμένο σε Python 2.7. Από την έκδοση 0.3.1 και έπειτα, έγινε μετάβαση στην Python 3.6. Το γραφικό περιβάλλον είναι γραμμένο σε HTML5 (HyperText Markup Language), CSS3 (Cascading Style Sheets) και JavaScript. Το αρχείο παραμετροποίησης που χρησιμοποιείται μέχρι και την τελευταία έκδοση, προκειμένου να μπορεί ο χρήστης, αν θέλει, να κλειδώσει το PiLock, είναι γραμμένο σε YAML (YAML Ain't Another Markup Language). Τέλος, τα σενάρια που χρησιμοποιούνται για την αρχική εγκατάσταση του PiLock στο RPi, είναι γραμμένα σε Shell. Το λογισμικό της εφαρμογής Android, καθώς επίσης και της εφαρμογής για Android Wear, είναι γραμμένα σε Java και XML (Cross Markup Language). Τα σενάρια ξεκλειδώματος (Unlock Scripts) είναι γραμμένα σε Python και Wiring, ένα Framework κατασκευασμένο για προγραμματισμό μικροελεγκτών, το οποίο χρησιμοποιείται για προγραμματισμό στο Arduino.

### 5.4.2 Βιβλιοθήκες/Frameworks που χρησιμοποιήθηκαν

To Business Logic του εξυπηρετητή του PiLock είναι υλοποιημένο σε Django. Το Django είναι ένα Free And Open Source Web Framework γραμμένο σε Python και συντηρείται από το Django Software Foundation (DSF)<sup>[36]</sup>. Χρησιμοποιεί την αρχιτεκτονική MVT (Model-View-Template). To Django είναι φτιαγμένο προκειμένου να καταστήσει εύκολη την ανάπτυξη πολύπλοκων ιστοσελίδων, που χρησιμοποιούν σύνδεση με βάσεις δεδομένων. Βασίζεται στην αρχή του Don't Repeat Yourself (DRY), που σημαίνει οτι είναι έτσι σχεδιασμένο, ώστε να μπορέσει να μειώσει τις επαναλήψεις κομματιών κώδικα, η την επανάληψη ορισμού λειτουργικότητας σε ένα λογισμικό<sup>[37]</sup>. Το Django παρέχει σύστημα αφαιρετικότητας βάσεων δεδομένων (Database Abstraction), που σημαίνει οτι ο τελικός κώδικας είναι συμβατός με μια πληθώρα συστημάτων βάσεων δεδομένων (sqlite, MySQL, PostgreSQL). Αυτό καθιστά την μετάβαση του

συστήματος (αν χρειαστεί), σε κάποιο νέο σύστημα βάσης δεδομένων εύκολη, καθώς δεν χρειάζεται να αλλάξει ο κώδικας.

Ο πίνακας διαχείρησης του PiLock (AdminCP), χρησιμοποιεί το Bootstrap 3 framework προκειμένου να εξασφαλίσει ομαλή έμφάνιση των οπτικών στοιχείων του σε όλα τα πιθανά μεγάλη οικονόμων. Το Bootstrap δημιουργήθηκε από τον Mark Otto και τον Jacob Thornton, που εργάζονταν ως προγραμματιστής και designer, αντίστοιχα, στο Twitter το 2010 [19], όπου προέκυψε ανάγκη για ενοποίηση των εργαλείων που χρησιμοποιούνταν προκειμένου να είναι λιγότερο δαπανηρή η συντήρηση<sup>[20]</sup>. Το Twitter είναι ένα Free and Open Source έργο, και διανείμεται υπό την άδεια MIT.

Προκειμένου να μπορούν να εκτελεστούν κάποιες λειτουργίες του πίνακα διαχείρησης, όπως το ξεκλείδωμα απευθείας από τον πίνακα διαχείρησης, καθώς επίσης και κάποια οπτικά εφέ, χρησιμοποιήθηκε η βιβλίοισθήκη jQuery. Η jQuery είναι μια βιβλιοισθήκη γραμμένη σε Javascript και χρησιμοποιείται προκειμένου να απλοποιήσει την διαδικασία προγραμματισμού στο επίπεδο του πελάτη (Client-Side Scripting)<sup>[21]</sup>.

#### 5.4.3 Προγραμματιστικά Εργαλεία που Χρησιμοποιούνται

Καθ' όλη την διαδικασία ανάπτυξης του PiLock, χρησιμοποιήθηκαν διάφορα **Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης (Integrated Development Environments, IDE)**, τα οποία, κάποια εξ' αυτών αποτελούν λογισμικό ανοικτού κώδικα. Παρατίθενται παρακάτω.

##### PyCharm - JetBrains

Το PyCharm είναι ένα προγραμματιστικό περιβάλλον στοχευμένο στην ανάπτυξη λογισμικού με την γλώσσα Python. Αναπτύσσεται από την τσεχική εταιρία JetBrains<sup>[?]</sup> και είναι γραμμένο σε Java και Python. Διατίθενται 2 εκδόσεις του Pycharm: Το PyCharm Community, που είναι δωρεάν και ανοικτού κώδικα λογισμικό (υπόκειται, συγκεκριμένα, στην άδεια Apache<sup>[?]</sup>), και το PyCharm Professional, το οποίο διατίθεται επι πληρωμής και είναι κλειστού κώδικα. Το PyCharm Professional παρέχει επιπρόσθετη λειτουργικότητα από αυτή του Community. Και οι δύο εκδόσεις είναι συμβατές με Windows, Linux και macOS.

Το PyCharm περιέχει λειτουργικότητα ανάλυσης κώδικα, γραφικό αποσφαλματωή (graphical debugger), και ενσωμάτωση πολλών εργαλείων διαχείρησης εκδόσεων (βλ. 5.3.1 Version Control), προκειμένου να ελαττωθεί η χρήση αυτών των προγραμμάτων εκτός του περιβάλλοντος εργασίας.

Το PyCharm επιλέχτηκε κατά την ανάπτυξη του PiLock Server χυρίως λόγω του γραφικού αποσφαλματωή, της ανάλυσης κώδικα και της ενσωμάτωσης εργαλείων διαχείρησης εκδόσεων. Από την έκδοση 0.3.1 και έπειτα, χρησιμοποιήθηκε επίσης η λειτουργία Unit Testing που παρέχει προκειμένου να υλοποιηθεί μηχανισμός αυτοματοποιημένου ελέγχου εγκυρότητας κώδικα (βλ. ?? ??).

##### Android Studio

Για την ανάπτυξη της εφαρμογής Android και αργότερα, από την έκδοση 0.3.0 και μετά, για την ανάπτυξη της εφαρμογής για Android Wear, χρησιμοποιήθηκε το Android Studio. Το Android Studio αποτελεί το επίσημο περιβάλλον ανάπτυξης εφαρμογών για Android συστήματα. Κατασκευάζεται από την Google σε συνεργασία

με την JetBrains και είναι βασισμένο πάνω στο IntelliJ IDEA της JetBrains. Είναι γραμμένο σε Java και Kotlin.

To Android Studio αποτελεί αντικαταστάτη των Εργαλείων Ανάπτυξης Android του Eclipse (Android Development Tools, ADT), τα οποία χρησιμοποιούνταν για ανάπτυξη σε Android μέχρι και το 2015<sup>[?]</sup>.

## Άλλα Εργαλεία, Text Editors

**Sublime Text** To Sublime Text είναι ένας επεξεργαστής κειμένου γραμμένος σε C++ και Python από τον Jon Skimmer και τον Will Bond. Υποστηρίζει μια πληθώρα γλωσσών προγραμματισμού και με λειτουργικότητα όπως η "Goto Anything", η οποία υποστηρίζει γρήγορη πλοήγηση σε οποιοδήποτε σημείο του κώδικα, σε διάφορα αρχεία, σύμβολα ή γραμμές επιλέγεται από πολλούς προγραμματιστές ανά τον πλανήτη.

Στο PiLock χρησιμοποιήθηκε κατά την ανάπτυξη μέρους του γραφικού περιβάλλοντος του πίνακα διαχείρησης συγκεκριμένα για την δημιουργία και επεξεργασία των αρχείων HTML5, CSS3 και JavaScript που τον αποτελούν.

**GNU Nano** O GNU Nano είναι ένας επεξεργαστής κειμένου που χρησιμοποιεί γραφική διεπαφή γραμμής εντολών (Command Line Graphical User Interface) και υπάρχει προεγκατεστημένος στα περισσότερα συστήματα είδους Unix. Είναι φτιαγμένος έτσι ώστε να μπορεί να εξομοιώνει τον Pico Editor όσο το δυνατόν καλύτερα και παράλληλα παρέχει παραπάνω λειτουργικότητα από αυτόν<sup>[?]</sup>. Είναι δωρεάν και ανοικτού κώδικα λογισμικό και υπόκειται στην άδεια GNU General Public Licence (GPL).

Κατά την ανάπτυξη του PiLock, το GNU Nano χρησιμοποιήθηκε προκειμένου να γίνουν δοκιμές και αποσφαλμάτωση του PiLock Server, ενόσω είναι σε λειτουργία πάνω στο Raspberry Pi, καθώς είναι προσβάσιμο από απομακρυσμένο τερματικό, μέσω SSH.

**git-cola** To git-cola είναι ένα δωρεάν και ανοικτού κώδικα γραφικό περιβάλλον διαχείρησης αποθετηρίων Git, ανεπτυγμένο από τον David Aguilar σε Python, και χρησιμοποιεί την βιβλιοθήκη PyQt για την κατασκευή του γραφικού περιβάλλοντός του. Χρησιμοποιείται προκειμένου να καταστεί πιο εύκολη η διαχείρηση ενός αποθετηρίου git, παρέχοντας μεγάλο μέρος των λειτουργιών του git μέσω του εύχρηστου γραφικού περιβάλλοντός του. Υποστηρίζει λειτουργίες όπως ευανάγνωση προβολή του ιστορικού του αποθετηρίου, εύχρηστη λειτουργία δημιουργίας νέων commits, και πολλές άλλες. Υπόκειται στην άδεια GNU General Public Licence (GPL), version 2<sup>[22]</sup>.

# Κεφάλαιο 6

## Χρονοδιάγραμμα Κυκλοφορίας Εκδόσεων

Το PiLock κυκλοφορεί σε συγκεκριμένες εκδόσεις. Η κάθε έκδοση μπορεί να παρέχει είτε αυξημένη λειτουργικότητα, είτε αλλαγές προκειμένου να βελτιστοποιήσει ο ήδη υπάρχον κώδικας. Υπάρχουν 2 τύπου εκδόσεις, Minor (μικρές), Major (μεγάλες). Οι μικρές εκδόσεις συνεισφέρουν μικρές αλλαγές στην λειτουργικότητα του PiLock, και συνήθως αποτελούνται από βελτιστοποιήσεις στον κώδικα και διορθώσεις bugs. Οι μεγάλες εκδόσεις συνεισφέρουν μεγάλες αλλαγές στην λειτουργικότητα είτε του πελάτη, είτε του εξυπηρετητή και συνήθως περιέχουν πολλά bugfix.

Στις μεγάλες εκδόσεις γίνεται αλλαγή στο νούμερο ανάμεσα στην 1η και την 2η υποδιαστολή του κωδικού έκδοσης. Στις μικρές εκδόσεις αλλάζει το νούμερο μετά την 2η υποδιαστολή, από αριστερά. Συγκεκριμένα, με κάθε νέα ενημέρωση, γίνεται αύξηση κατά ένα του αντίστοιχου αριθμού. Αν είναι μεγάλη έκδοση, πρέπει να γίνει επαναφορά του αριθμού μικρής έκδοσης στο μηδέν (0).

Παρακάτω παρατίθενται οι εκδόσεις από την δημιουργία του PiLock, μέχρι τώρα. Έχουν επίσης αναρτηθεί συνοπτικά διαγράμματα των κλάσεων που χρησιμοποιήθηκαν, στο Παράρτημα A.1, καθώς επίσης και αναλυτικά διαγράμματα περιπτώσεων χρήσης (Use Case) στο Παράρτημα A.2.

### 6.1 0.1.0

Ημερομηνίες ανάπτυξης:

Έναρξη: 1η Απριλίου 2017

Κυκλοφορία: 17 Μαΐου 2017

Η έκδοση 0.1.0 ήταν η 1η έκδοση που βγήκε από την πρώτη σύλληψη του PiLock και μετά και παρείχε την βασική λειτουργικότητα, αρκετή για να λειτουργήσει το κύκλωμα ξεκλειδώματος.

Μέχρι τότε, η εφαρμογή πελάτη για Android είχε πολλά προβλήματα κατά την λειτουργία της, με πολλά crash, αλλά το βασικό σύστημα ξεκλειδώματος είχε υλοποιηθεί και δούλευε.

## 6.2 0.2.0

Ημερομηνίες ανάπτυξης:

Έναρξη: 29 Ιουλίου 2017

Κυκλοφορία: 18 Αυγούστου 2017

Η έκδοση 0.2.0 αποτελεί την πρώτη δημόσια διαθέσιμη έκδοση του PiLock. Σε αυτή την έκδοση έγινε η προσθήκη του πίνακα διαχείρησης στην πρωτογενή μορφή του (πριν την κυκλοφορία της έκδοσης αυτής, οι χρήστες έπρεπε να χρησιμοποιούσαν τον συμπεριλαμβανόμενο στο Django πίνακα διαχείρησης).

Επίσης, έγινε μια τεράστια αλλαγή στον οπτικό σχεδιασμό της εφαρμογής. Το κύριο χρώμα της εφαρμογής mobile άλλαξε σε ρόζ (#790022) και έγινε προσθήκη ενός νέου λογότυπου φτιαγμένο από τον Δημήτρη Τζιλβάκη.

Στον πίνακα διαχείρησης προστέθηκε δυνατότητα διαχείρησης χρηστών, συγκεκριμένα, λειτουργία προσθήκης νέων και διαγραφής ήδη υπάρχοντων χρηστών και προφίλ συσκευών, καθώς επίσης και ένα ημερολόγιο ξεκλειδωμάτων και συνδέσεων στο σύστημα, για λόγους ασφαλειας.

## 6.3 0.3.0

Ημερομηνίες ανάπτυξης:

Έναρξη: 7 Σεπτεμβρίου 2017

Κυκλοφορία: 11 Νοεμβρίου 2017

Στην έκδοση 0.3.0 προστέθηκε πλήθος νέων λειτουργιών που βοηθούν στην διαχείρηση και στην χρήση του PiLock. Προστέθηκε, συγκεκριμένα δυνατότητα ξεκλειδώματος χωρίς την χρήση PIN, ένα νέο σύστημα ειδοποίησεων προσβάσιμο από τον πίνακα διαχείρησης από όπου ο χρήστης μπορεί να δει διάφορα μηνύματα σχετικά με την κατάσταση του PiLock, καθώς επίσης και δυνατότητα ξεκλειδώματος μέσω του πίνακα διαχείρησης.

Στην νέα λειτουργικότητα εντάσσεται επίσης η εφαρμογή για Android Wear συσκευές, που επιτρέπει στον χρήστη, εφόσον είναι συνδεδεμένο το SmartPhone του με το Smartwatch του, να γίνει ξεκλείδωμα, μέσω του Android Wear Smartwatch του.

Τέλος, προστέθηκε hashing στην βάση δεδομένων προκειμένου να αποτρέψει προβολή των κλειδιών πρόσβασης σε περίπτωση που υπάρξει μη εξουσιοδοτημένη πρόσβαση στην βάση δεδομένων<sup>[23]</sup> (βλ 8.4.1 Στάδιο Εισόδου - Login).

## 6.4 0.3.1

Ημερομηνίες ανάπτυξης:

Έναρξη: 25 Φεβρουαρίου 2018

Κυκλοφορία: 4 Μαρτίου 2018

Στην Minor έκδοση 0.3.1 του PiLock, έγινε ενημέρωση του κώδικα ώστε να χρησιμοποιείται πλέον η Python 3.6, καθώς η Python 2.7 φτάνει σιγά σιγά στο τέλος ζωής της<sup>[?]</sup> και οι νέες ενημερώσεις για το Django ωστε γίνονται μόνο στην έκδοση 3 της Python<sup>[?]</sup>. Εγιναν πολλές διορθώσεις στον κώδικα του Server, προκειμένου να λειτουργεί καλύτερα, ενεργοποιήθηκε η χρήση του venv και ενεργοποιήθηκε η δυνατότητα αυτοματοποιημένων ελέγχων εγκυρότητας κώδικα μέσω Continuous Integration (CI).

Από άποψης λειτουργικότητας, έγινε προσθήκη σελιδοποίησης στο ημερολόγιο πρόσβασης του συστήματος, προκειμένου να φαίνονται οι απόπειρες πρόσβασης πιο καθαρά.

# Κεφάλαιο 7

## Σενάρια Ξεκλειδώματος - PiLockUnlockScripts

Τα σενάρια (scripts) ξεκλειδώματος του PiLock, γνωστά ως PiLockUnlockScripts, αποτελούν κομμάτι του λογισμικού του εξυπηρετητή. Συνδέονται με το Django Project του εξυπηρετητή μέσω ενός Git Submodule που έχει τοποθετηθεί στο main app.

Η δέσμη των σεναρίων αποτελείται από ένα σενάριο γραμμένο σε Python και ένα πρόγραμμα Arduino, σε περίπτωση που ο χρήστης επιθυμεί να χρησιμοποιήσει Arduino για να πραγματοποιεί ξεκλειδωμα (βλ. 4.2.2 Σύνδεση με την χρήση Arduino).

```
1 import serial
2 from time import sleep
3 import RPi.GPIO as GPIO
4
5 # Mode switch
6 # 0 -> GPIO Unlock
7 # 1 -> Arduino Assisted Unlock (default before 0.3.1)
8 MODE = 0
9 # The GPIO pin used for the switch. Refer to your RPi's PinOut
# diagram.
10 # Used only when MODE = 0
11 SW_PIN = 18
12
13 def unlock():
14     if MODE == 0:
15         GPIO.setmode(GPIO.BCM)
16         GPIO.setwarnings(False)
17         GPIO.setup(SW_PIN, GPIO.OUT)
18         GPIO.output(SW_PIN, GPIO.HIGH)
19         sleep(5)
20         GPIO.output(SW_PIN, GPIO.LOW)
21         GPIO.cleanup()
22     else:
23         ser = serial.Serial(port="/dev/ttyACM0", baudrate=9600)
24         ser.isOpen()
25         sleep(2) # Wait for the arduino to be ready.
26         ser.write(b"ENABLE")
27         ser.close()
28
29 if __name__ == "__main__":
30     unlock()
```

Listing 7.1: Σενάριο Ξεκλειδώματος Python (unlock.py)

## 7.1 Ξεκλείδωμα μέσω GPIO

Ο προεπιλεγμένος τρόπος ξεκλειδώματος, από την έκδοση 0.3.1 και έπειτα, είναι το ξεκλείδωμα μέσω των δεκτών GPIO του Raspberry Pi. Αφότου συνδεθεί το Raspberry Pi με ένα συμβατό Relay Module (βλ. 4.2.1 Σύνδεση χωρίς την χρήση Arduino), μπορεί να εκτελεστεί το προεπιλεγμένο σενάριο ξεκλειδώματος (7.1 Σενάριο Ξεκλειδώματος Python (unlock.py)).

Προκειμένου να οριστεί ο τρόπος ξεκλειδώματος, πρέπει να αλλάξει η μεταβλητή MODE σε 0, αν πρόκειται να χρησιμοποιηθεί το GPIO ως μέθοδος ξεκλειδώματος, και οτιδήποτε άλλο αν πρόκειται να χρησιμοποιηθεί Arduino για ξεκλείδωμα.

Αν πρόκειται να χρησιμοποιηθεί το GPIO ως μέθοδος ξεκλειδώματος, πρέπει να γίνει επίσης τροποποίηση της μεταβλητής SW\_PIN, προκειμένου να οριστεί το GPIO Pin το οποίο θα χρησιμοποιηθεί για το Relay. Στο παρόδειγμα Figure 4.2, χρησιμοποιείται το Pin #18, κατά BCM<sup>[26]</sup>.

Στην γραμμή 15, ορίζεται η λειτουργία αρίθμησης ως BCM και στην 16 γίνεται απενεργοποίηση των προειδοποιήσεων του συστήματος GPIO. Στην γραμμή 17 γίνεται ορισμός του GPIO Pin που ορίστηκε από την μεταβλητή SW\_PIN ως έξοδος. Στις γραμμές 18, 19 και 20 πραγματοποιείται το ξεκλείδωμα. Συγκεκριμένα, στέλνεται ψηφιακό σήμα 1 στο Relay Module μέσω του Pin εξόδου που έχει οριστεί. Κατά την συγκεκριμένη χρονική στιγμή, ο μηχανισμός ξεκλειδώματος της πόρτας (βλ. 3 Συστήματα Ελέγχου Πρόσβασης Πολυκατοικιών/Σπιτιών) δέχεται ρεύμα και ενεργοποιείται, δηλαδή μπορεί κάποιος, αν σπρώξει την πόρτα, να την ανοίξει. Το σύστημα παραμένει στην συγκεκριμένη κατάσταση για 5 δευτερόλεπτα (γραμμή 19) και έπειτα ξαναγυρνά το Pin εξόδου σε 0, και απενεργοποιείται ο μηχανισμός ξεκλειδώματος (γραμμή 20). Τέλος, γίνεται εκκαθάριση των GPIO (γραμμή 21). Το ξεκλείδωμα ολοκληρώθηκε.

## 7.2 Ξεκλείδωμα μέσω Arduino

Προκειμένου να καταστεί δυνατόν να πραγματοποιούνται ξεκλειδώματα μέσω Arduino, θα πρέπει πρώτα να γίνει μεταφόρτωση του ακόλουθου σεναρίου προγραμματισμού στο Arduino, μέσω του Arduino IDE:

```
1 // The pin the relay is attached to.
2 const int RELAY_PIN = 2;
3 // How many seconds to keep the relay turned on.
4 const int DELAY = 5;
5
6 void setup() {
7     // Setup everything...
8     Serial.begin(9600);
9     pinMode(RELAY_PIN, OUTPUT);
10    pinMode(LED_BUILTIN, OUTPUT);
11 }
12
13 void loop() {
14     // The Relay is active low, so write high to it in order to disable
15     // it.
16     digitalWrite(RELAY_PIN, HIGH);
17     digitalWrite(LED_BUILTIN, LOW);
```

```

18 // If the serial port is available , read each character and
19 concatenate it to a string .
20 if (Serial.available() > 0){
21     String content = "";
22     char character;
23
24     while( Serial.available() ) {
25         character = Serial.read();
26         content.concat(character);
27         delay(10); // Add a 10ms delay in order to receive the
28         characters correctly .
29     }
30
31     // Finally , check the string , if it reads "ENABLE" , turn on the
32     relay , wait for 5 seconds , then turn it off .
33     if (content == "ENABLE") {
34         digitalWrite(RELAY_PIN, LOW);
35         digitalWrite(LED_BUILTIN, HIGH);
36         delay(DELAY * 1000);
37     }
38 }
39 }
```

Listing 7.2: Σενάριο Ξεκλειδώματος Arduino

Προκειμένου να γίνει αντιληπτό το παραπάνω σενάριο, είναι αναγκαίο να αναλυθεί η ανάγκη χρήσης της σειριακής θύρας του Arduino. Όλες οι πλακέτες Arduino έχουν τουλάχιστον μία Hardware-Based σειριακή θύρα. Αυτή χρησιμοποιείται προκειμένου να γίνει ανταλλαγή δεδομένων με κάποιο υπολογιστή ή με κάποια άλλη συσκευή συνδεδεμένη στο Arduino<sup>[27]</sup>. Στο PiLock, η σειριακή θύρα του Arduino χρησιμοποιείται για να ενεργοποιήσει το Arduino το Relay Module, την στιγμή που είναι επιθυμητό. Συγκεκριμένα, όταν η σειριακή θύρα λάβει την λέξη "ENABLE", γίνεται ενεργοποίηση του Relay Module (γραμμές 30-34). Στις γραμμές 23-27 γίνεται σχηματισμός της λέξης με έναν χαρακτήρα την φορά, καθώς λαμβάνεται από την σειριακή θύρα.

Και πάλι χρησιμοποιείται η ίδια λογική ενεργοποίησης του Relay Module με αυτή που χρησιμοποιείται όταν γίνεται ξεκλειδωμα μέσω GPIO, αλλά σε αυτή την περίπτωση γίνεται αντίστροφα (0 για ενεργοποίηση, 1 για απενεργοποίηση), καθώς το σενάριο είναι γραμμένο για Relays που δουλεύουν με λογική Active Low. Μπορεί να γίνει τροποποίηση του σεναρίου για να λειτουργεί με Active High Relay Modules, αλλάζοντας τις λέξεις HIGH σε LOW και αντίστροφα.

Την αποστολή της λέξης στο Arduino αναλαμβάνει το σενάριο ξεκλειδώματος unlock.py που αναφέρθηκε προηγουμένως (7.1 Σενάριο Ξεκλειδώματος Python (unlock.py)). Συγκεκριμένα, στην γραμμή 23 γίνεται ενεργοποίηση της σειριακής θύρας του Raspberry Pi. Έπειτα ακολουθεί έλεγχος για εαν η θύρα έχει ανοίξει, και γίνεται καθυστέρηση 2 δευτερολέπτων, προκειμένου να μπορέσει το Arduino να προετοιμαστεί, από την πλευρά του. Τέλος, στην γραμμή 26 στέλνεται στην σειριακή θύρα η λέξη "ENABLE" και γίνεται κλείσιμο της σειριακής θύρας στην γραμμή 27.

### 7.3 Χρήση των σεναρίων ξεκλειδώματος

Προκειμένου να μπορέσουν να εκτελεστούν τα σενάρια ξεκλειδώματος, θα χρειαστεί είτε κάποιος να καλέσει το σενάριο unlock.py απευθείας, είτε να εισάγει την συνάρτηση unlock στον δικό του κώδικα και να την χρησιμοποιήσει:

```
1   from .PiLockUnlockScripts.unlock import unlock  
2   unlock()
```

Για να μπορέσουν να τρέξουν τα σενάρια ξεκλειδώματος θα πρέπει να εισαχθεί ο χρήστης Unix που πρόκειται να τα εκτελέσει, στις ομάδες dialout και gpiο, προκειμένου να πάρει τα κατάλληλα δικαιώματα εκτέλεσης. Στην ομάδα dialout πρέπει να εισαχθεί προκειμένου να αποκτήσει πρόσβαση στην σειριακή θύρα του Raspberry Pi και στην ομάδα gpiο πρέπει να εισαχθεί προκειμένου να μπορέσει να διαχειριστεί τα GPIO headers. Η διαδικασία αυτή θα στο κεφάλαιο 12 Εγκατάσταση και ρύθμιση του PiLock Server.

Τα σενάρια που αναφέρθηκαν σε αυτό το κεφάλαιο, μπορούν να βρεθούν στο επίσημο αποθετήριο τους, στο GitHub:

<https://github.com/kerk12/PiLockUnlockScripts>

# Κεφάλαιο 8

## Ο Διακομιστής του PiLock - PiLock Server

Στο παρόν κεφάλαιο θα αναλυθεί η δομή του διακομιστή του PiLock, και παράλληλα θα εξηγηθεί ο τρόπος λειτουργίας του. Όπως ειπώθηκε στην παράγραφο 5.4.2 Βιβλιοθήκες/Frameworks που χρησιμοποιήθηκαν, ο διακομιστής του PiLock χρησιμοποιεί το Django, ένα Web Framework γραμμένο σε Python, προκειμένου να χειρίζεται το Business Logic με το οποίο λειτουργεί.

### 8.1 Αρχιτεκτονική MTV

Πρωτού αναλυθεί το Business Logic του διακομιστή, είναι αναγκαίο να αναλυθεί η Αρχιτεκτονική του Framework του οποίου χρησιμοποιείται. Η αρχιτεκτονική του Django, γνωστή ως MTV, αποτελείται από 3 ξεχωριστά εξαρτήματα, τα οποία συνεργάζονται προκειμένου να παραχθεί η τελική σελίδα, η οποία παραδίδεται στον πελάτη:

- Το **Μοντέλο (Model, M)** είναι μία αναπαράσταση των δεδομένων τα οποία χρησιμοποιεί και επεξεργάζεται ο διακομιστής. Το μοντέλο δεν αποτελεί τα ίδια τα δεδομένα, αλλά μια διεπαφή μέσω της οποίας ο διακομιστής μπορεί να αντλήσει και να χρησιμοποιήσει δεδομένα. Για παράδειγμα, αν δημιουργηθεί μια εφαρμογή για επαφές, η **Επαφή**, μαζί με όλα της τα στοιχεία (Όνοματεπώνυμο, Τηλέφωνο, κτλ...) θα αποτελούσε ένα μοντέλο.
- Το **Πρότυπο (Template, T)**, το οποίο αποτελεί το στρώμα παρουσίασης (Presentation Layer) των δεδομένων. Καθορίζει την μορφή με την οποία θα παρουσιαστούν τα δεδομένα στον πελάτη. Καθώς το Django πρόκειται για ένα Web Framework, το Template σχηματίζει τις τελικές σελίδες Web που θα παραδώσει ο εξυπηρετητής στον πελάτη.
- Την **Προβολή (View, V)**, η οποία αποτελεί την επιχειρησιακή λογική (Business Logic) του εξυπηρετητή, δηλαδή τον τρόπο χειρισμού και επεξεργασίας των δεδομένων πριν να παραδοθούν στο εκάστοτε Template. Αποτελεί το στρώμα ενδιάμεσα στο Model και στο Template.

## 8.2 Django Apps

Ως πρώτο στρώμα για ένα οποιοδήποτε έργο στο Django αποτελεί το Project. Το Project αποτελείται από διάφορες συλλογές κώδικα γνωστές ως **Εφαρμογές (Django Apps)**, και ορίζει το περιβάλλον εκτέλεσης του κώδικα. Περιέχει όλες τις ρυθμίσεις απαραίτητες για να εκτελεστεί επιτυχώς ένα έργο. Μέσω των Apps οργανώνεται η επιχειρισιακή λογική ενός έργου σε πολλά, διαχριτά κομμάτια, που είτε εξαρτόνται, είτε όχι μεταξύ τους.

Το PiLock αποτελείται από 2 Django Apps: Την ”main” (κύρια εφαρμογή), η οποία, μέσω της επιχειρισιακής λογικής τής αποτελεί την κύρια **Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming Interface, API)**, υπεύθυνη για την επικοινωνία με την εφαρμογή πελάτη και για την λειτουργία του συστήματος ξεκλειδώματος, και την ”AdminCP” (Administration Control Panel), η οποία αποτελεί τον πίνακα διαχείρησης του PiLock.

## 8.3 Μοντέλα που Ορίστηκαν/Χρησιμοποιούνται

Όπως αναφέρθηκε πριν (βλ. 8.1 Αρχιτεκτονική MTV), η αρχιτεκτονική του Django απαιτεί να οριστούν κάποια μοντέλα απεικόνισης των δεδομένων που θα επεξεργάζονται από τον διακομιστή. Κατά την ανάπτυξη του PiLock, ορίστηκαν τα παρακάτω μοντέλα:

- **Profile (Device Profile):** Αποτελεί το προφίλ μιας κινητής συσκευής με εξουσιοδότηση ξεκλειδώματος. Αποτελείται από τα εξής πεδία:
  - **user:** Ο χρήστης της πλατφόρμας του οποίου του ανήκει η συσκευή αυτή. ”Δένεται” με έναν συγκεκριμένο χρήστη της πλατφόρμας μέσω του Πρωτεύοντος Κλειδιού του (Primary Key, PK), έτσι ώστε ο εξουσιοδοτημένος χρήστη να μπορεί να έχει αποκλειστικά μία εξουσιοδοτημένη συσκευή. Η σχέση αυτή ονομάζεται One-To-One.
  - **authToken (Authorization Token):** Ένα τυχαία παραγόμενο αλφαριθμητικό, μήκους 50 χαρακτήρων που δημιουργείται όταν δημιουργείται και το προφίλ συσκευής. Αποτελεί τεκμήριο οτι η συγκεκριμένη συσκευή είναι εξουσιοδοτημένη. Στην ΒΔ αποθηκεύεται σε Hashed μορφή (SHA512).
  - **pin (Personal Identification Number, PIN):** Ένας βψήφιος αριθμητικός προσωπικός κωδικός. Δίδεται στον χρήστη την στιγμή που δημιουργείται το προφίλ συσκευής και μπορεί, αν θελήσει να τον αλλάξει. Στην ΒΔ αποθηκεύεται σε Hashed μορφή (SHA512). Μπορεί να είναι κενό το πεδίο, σε περίπτωση που ο χρήστης χρησιμοποιεί ξεκλείδωμα χωρίς PIN.
  - **wearToken (Wear Token):** Ένα τυχαία παραγόμενο αλφαρηθμητικό, μήκους 32 χαρακτήρων που δημιουργείται την στιγμή που ο χρήστης θα συγχρονίσει το Android Wear Smartwatch του με το PiLock. Στην ΒΔ αποθηκεύεται σε Hashed μορφή (SHA512). Μπορεί να είναι κενό το πεδίο, αν ο χρήστης δεν συγχρονίσει το Smartwatch του.
- **AccessAttempt (Access Attempt):** Αποτελεί μια καταγραφή απόπειρας εισόδου/ξεκλειδώματος. Χρησιμοποιείται από το ημερολόγιο πρόσβασης του πίνακα διαχείρησης. Αποτελείται από τα εξής πεδία:

- **usernameEntered (Entered Username):** Το όνομα χρήστη που πληκτρολόγησε ο χρήστης κατά την είσοδό του στην πλατφόρμα. Αλφαριθμητικό, μέγιστου μήκους 130 χαρακτήρων. Αν το σύστημα δεν καταφέρει να εντοπίσει κάποιο όνομα χρήστη, παραμένει κενό.
- **is\_unlock\_attempt:** Boolean πεδίο. Παίρνει την τιμή True αν η απόπειρα εισόδου ήταν προκειμένου να γίνει ξεκλείδωμα, False αν όχι.
- **successful:** Boolean πεδίο. Παίρνει την τιμή True αν έγινε επιτυχής εξαρίβωση στοιχείων και παραχωρήθη η πρόσβαση, False αν δεν παραχωρήθηκε πρόσβαση.
- **ip:** Διεύθυνση IP του πελάτη, κατά την απόπειρα εισόδου.
- **datetime:** Ημερομηνία και ώρα που πραγματοποιήθηκε η απόπειρα εισόδου στο σύστημα.

- **Notification:** Αποτελεί μια ειδοποίηση που εμφανίζεται στον πίνακα διαχείρησης, προκειμένου να ειδοποιηθεί ο χρήστης για κάποιο σημαντικό ή μή ζήτημα σχετικό με την εγκατάσταση του PiLock. Αποτελείται από τα εξής πεδία:

- **type:** Ο τύπος της ειδοποίησης. Μπορεί, μέχρι και την τελευταία έκδοση (0.3.1, κατά τον χρόνο συγγραφής της παρούσας εργασίας), να πάρει μία εκ των παρακάτω τιμών: "DEBUG", αν είναι ενεργοποιημένο το Debug Mode, "UPDATE", αν υπάρχει διαθέσιμη κάποια ενημέρωση για το PiLock, και "SEC", αν υπάρχει κάποιο ζήτημα ασφάλειας (δεν χρησιμοποιείται ακόμα, θα χρησιμοποιηθεί σε μία επόμενη έκδοση).
- **text:** Το κείμενο που θα εμπεριέχεται στη ειδοποίηση.
- **created:** Ημερομηνία και ώρα δημιουργίας της ειδοποίησης. Χρησιμοποιείται σε περίπτωση που χρειαστεί να γίνει αποσφαλμάτωση του συστήματος.

Αξίζει να αναφερθεί οτι ανάλογα με τον τύπο της ειδοποίησης, και εάν το πεδίο **text** παραμείνει κενό, η ειδοποίηση αυτόματα λαμβάνει ένα εκ των προκαθορισμένων κειμένων για τον συγκεκριμένο τύπο ειδοποίησης (βλ. ?? ??).

## 8.4 Σύστημα Εξουσιοδότησης Πρόσβασης

Για την σωστή λειτουργία του συστήματος ξεκλειδώματος του PiLock, ήταν αναγκαίο να σχεδιαστεί ένα σύστημα αυθεντικοποίησης ικανό να εξουσιοδοτεί τους χρήστες με ασφάλεια και να παρέχει προστασία από διάφορες πιθανές επιθέσεις. Το σύστημα εξουσιοδότησης λειτουργεί σε 2 στάδια: Το 1o στάδιο αποτελεί το στάδιο της εισόδου (Login Stage), μέσω του οποίου μία κινητή συσκευή εξουσιοδοτείται προκειμένου να μπορεί να ζητήσει ξεκλείδωμα (οι μη εξουσιοδοτημένες συσκευές δεν μπορούν να ζητήσουν ξεκλείδωμα, παρά μόνο αν περάσουν από το στάδιο της εισόδου πρώτα). Το 2o και τελευταίο στάδιο αποτελεί το στάδιο του ξεκλειδώματος, μέσω του οποίου μία εξουσιοδοτημένη συσκευή στέλνει κάποια διαπιστευτήρια που έχουν της δωθεί από το 1o στάδιο, εάν ήταν επιτυχής η είσοδος, προκειμένου ο Server να ενεργοποιήσει το Relay. Παρακάτω αναλύονται αναλυτικά τα στάδια.

Πρωτού αναλυθούν, καλό θα είναι να οριστούν κάποιοι όροι που θα χρησιμοποιηθούν παρακάτω:

- **Μη Εξουσιοδοτημένη Συσκευή:** Μια συσκευή που δεν διαθέτει τα διαπιστευτήρια που χρησιμοποιούνται για να ξεκλειδωθεί η πόρτα. Το σύστημα ξεκλειδώματος ΔΕΝ ενεργοποιείται, εφόσον η συσκευή στείλει το αίτημα ξεκλειδώματος.
- **Εξουσιοδοτημένη Συσκευή:** Μια συσκευή με όλα τα απαιτούμενα διαπιστευτήρια προκειμένου να ενεργοποιηθεί το σύστημα ξεκλειδώματος.
- **Μη εξουσιοδοτημένος χρήστης:** Ένας χρήστης που δεν διαθέτει στοιχεία πρόσβασης στην εγκατάσταση του PiLock, ή που έχει ξεχάσει τον Προσωπικό Αριθμό Αναγνώρισής (PIN) του.
- **Εξουσιοδοτημένος χρήστης:** Χρήστης που να έχει στοιχεία πρόσβασης στην εγκατάσταση του PiLock ή/και γνωρίζει το PIN του.

#### 8.4.1 Στάδιο Εισόδου - Login

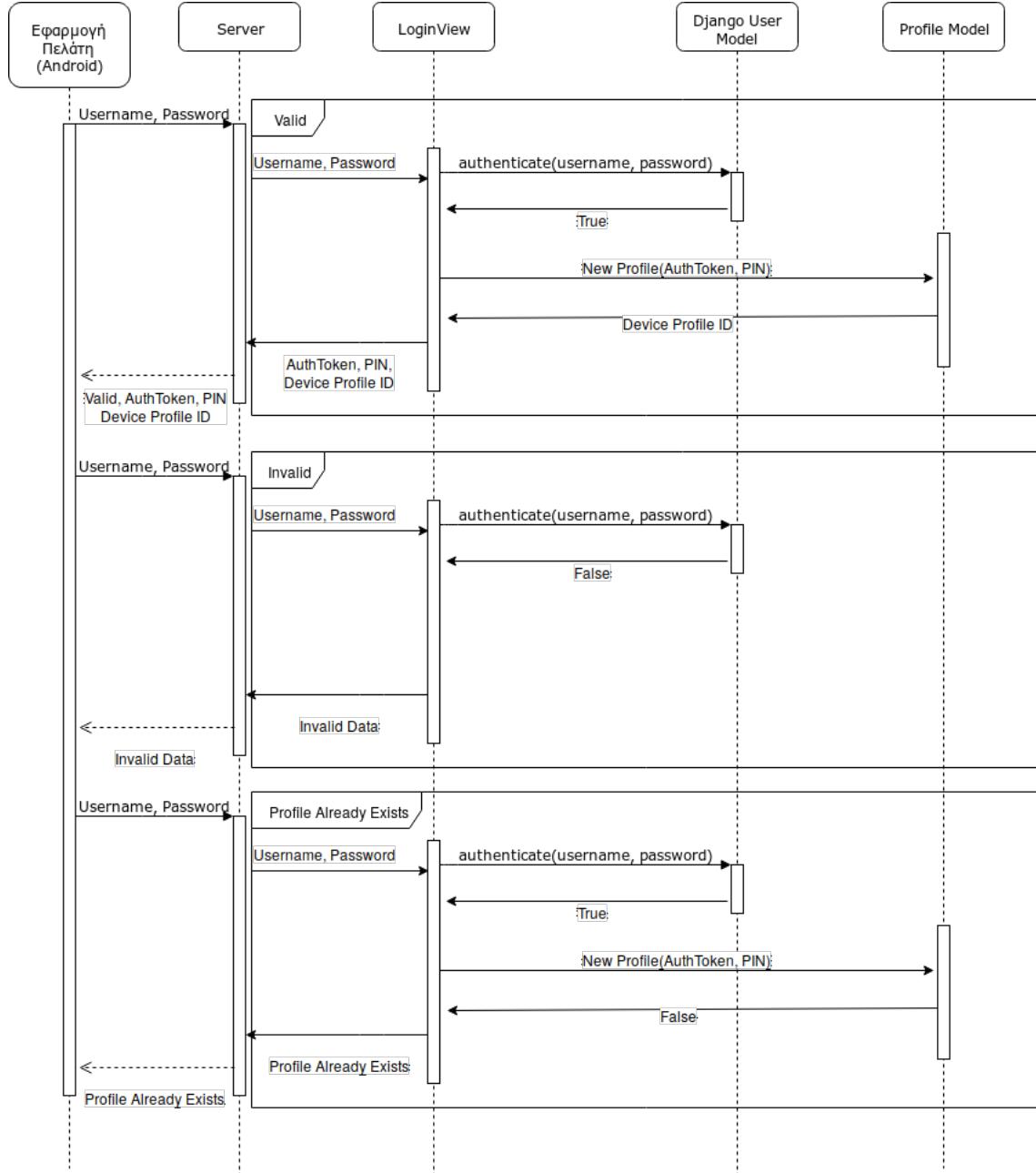
Όλοι οι χρήστες, προκειμένου να αποκτήσουν στοιχεία πρόσβασης στην εφαρμογή, πρέπει να γραφτούν στο σύστημα μέσω του διαχειριστή της εγκατάστασης. Αφότου ο διαχειριστής τους εγγράψει μέσω του πίνακα διαχείρησης, τους δίδεται ένα όνομα χρήστη (Username) και ένας κωδικός πρόσβασης (Password), τα οποία αποτελούν διαπιστευτήρια στοιχεία για την είσοδο τους στο σύστημα.

Κατά την δημιουργία των στοιχείων πρόσβασης του εκάστοτε χρήστη, **γίνονται κάποιοι ελέγχοι πολυπλοκότητας στον κωδικό πρόσβασης**, προκειμένου να διαπιστωθεί αν ο κωδικός πρόσβασης είναι αρκετά ασφαλής. Αυτοί είναι οι εξής: **Έλεγχος ελάχιστου μήκους**, προκειμένου ο κωδικός πρόσβασης να έχει ένα συγκεκριμένο ελάχιστο μήκος, **Έλεγχος ομοιότητας** με διάφορα άλλα στοιχεία του χρήστη που δώθηκαν κατά την εγγραφή (email, όνομα, επώνυμο, username), **έλεγχος συνηθισμένου κωδικού**, προκειμένου ο κωδικός να μην είναι στο 1.000.000 πιο συνηθείς κωδικούς πρόσβασης, και τέλος **έλεγχος μη αριθμητικού κωδικού**, προκειμένου να διαπιστωθεί αν ο κωδικός πρόσβασης είναι εξ'ολοκλήρου αριθμητικός. Αν δεν περάσει κάποιος από τους παραπάνω ελέγχους, η δημιουργία χρήστη απορρίπτεται και πρέπει να γίνει εισαγωγή νέου κωδικού πρόσβασης<sup>[24]</sup>.

Αυτά τα στοιχεία πρέπει να τα χρησιμοποιήσει, έπειτα, ο χρήστης προκειμένου να κάνει είσοδο μέσω της εφαρμογής Android. Με την πρώτη είσοδό τους στο σύστημα, αυτόματα δημιουργείται ένα προφίλ συσκευής στο οποίο **προστίθεται ένα τυχαίο τεκμήριο εξουσιοδότησης (Authorization Token, Auth Token)** και **ένα τυχαίο 6ψήφιο αριθμητικό PIN**. Εάν ο χρήστης ζητήσει να γίνεται ξεκλειδωματικώρις PIN, παραλαβείται η δημιουργία του PIN. Και τα δύο παραπάνω στοιχεία εξουσιοδότησης, καθώς επίσης και ο αριθμός προφίλ συσκευής, με βάση το πρωτεύον κλειδί του, **αποστέλλονται στον πελάτη και έπειτα γίνεται hashing τους** προκειμένου να αποφευχθεί υποκλοπή σε περίπτωση που κάποιος καταφέρει να αποκτήσει πρόσβαση στη ΒΔ. Το Hashing γίνεται με χρήση της συνάρτησης **SHA512**, από την βιβλιοθήκη `passlib` της Python. Χρησιμοποιείται SHA των 512 bits προκειμένου να αποφευχθούν όσο το δυνατόν περισσότερο πιθανές συγκρούσεις (collisions) μεταξύ αποτελεσμάτων. Από την στιγμή που αποσταλλούν στην εφαρμογή πελάτη, γίνεται κρυπτογράφηση και αποθήκευση του Auth Token μέσω του Android Keystore και εμφανίζεται το PIN για μία φορά στον χρήστη, προκειμένου να το σημειώσει, και κατόπιν απορρίπτεται για λόγους ασφαλείας. Πλέον η συσκευή του είναι εξουσιοδοτημένη. Σε αυτό το σημείο τελειώνει το 1ο στάδιο της εξουσιοδότησης.

Αν τα στοιχεία πρόσβασης του χρήστη (Username, Password) είναι λάθος, απορίπτεται η σύνδεση.

Σε περίπτωση που προσπαθήσει ένας χρήστης να ξαναεισέλθει και υπάρχει **ήδη** ένα προφίλ συσκευής αντιστοιχισμένο στον λογαριασμό του, δεν επιτρέπεται καμία ενέργεια (απορρίπτεται η είσοδος). Σε περίπτωση που ο χρήστης χρειαστεί να προσδέσει νέα συσκευή στο PiLock, θα πρέπει να διαχραφεί το ήδη υπάρχον προφίλ (μέσω του πίνακα διαχείρησης) και έπειτα να ξαναγίνει σύνδεση της συσκευής του.



Εικόνα 8.1: Στάδιο Εισόδου

#### 8.4.2 Στάδιο Ξεκλειδώματος - Unlock

Από την στιγμή που μια συσκευή αποκτήσει το δικό της προφίλ, και θεωρηθεί εξουσιοδοτημένη, μπορεί να αποστείλει αιτήματα ξεκλειδώματος.

Αρχικά, ο χρήστης πληκτρολογεί στην εφαρμογή Android το 6ψήφιο προσωπικό του PIN και αγγίζει το κουμπί ”Unlock”. Ξεκινά η διαδικασία ξεκλειδώματος. Το πρώτο βήμα είναι να ανακτηθεί το Auth Token από τα Shared Preferences της συσκευής (σε κρυπτογραφημένη μορφή) και να αποκρυπτογραφηθεί με χρήση του Android Key-store. Αφότου αποκρυπτογραφηθεί, αποστέλει η εφαρμογή Android το Auth Token, τον αριθμό προφίλ της συσκευής και το PIN που πληκτρολόγησε ο χρήστης, στον Server.

Ο Server λαμβάνει και τα τρία και πραγματοποιεί αναζήτηση για κάποιο προφίλ με αυτόν τον αριθμό προφίλ συσκευής που έλαβε. Αν υπάρχει, γίνεται επαλήθευση του Auth Token και του PIN που δώθηκαν. Αν επαληθεύονται επιτυχώς, γίνεται ενεργοποίηση του μηχανισμού ξεκλειδώματος και στέλνεται απάντηση (response) επιτυχίας στην συσκευή. Αν δεν επαληθεύονται στέλνεται απάντηση αποτυχίας.

Αν δεν υπάρχει προφίλ με αυτό τον αριθμό, και πάλι η διαδικασία σταματάει με αποτυχία. Και στις δύο περιπτώσεις (επιτυχίας, αποτυχίας), γίνεται καταγραφή της απόπειρας πρόσβασης στο ημερολόγιο πρόσβασης του συστήματος.

#### 8.4.3 Ξεκλείδωμα μέσω Android Wear - Wear Unlock

Προκειμένου να γίνει εφικτό να γίνονται αιτήματα ξεκλειδώματος μέσω Android Wear κατά βούληση, θα πρέπει να γίνει πρώτα ένας ”συγχρονισμός” με την Android Wear συσκευή. Προκειμένου να γίνει αυτός ο συγχρονισμός, απαιτείται η συσκευή να είναι εξουσιοδοτημένη (να έχει περάσει επιτυχώς το στάδιο εισόδου).

Αφότου ο χρήστης πληκτρολογήσει το PIN του στην κεντρική ουδόνη ξεκλειδώματος της εφαρμογής Android, μέσω ενός πλήκτρου που υπάρχει στο μενού του Action Bar, μπορεί να γίνει συγχρονισμός με την Android Wear συσκευή του. Με το χλικ σε αυτό το πλήκτρο, αποστέλλεται αιτημα γέννησης τεκμηρίου εξουσιοδότησης Android Wear (βλ. 8.3 Μοντέλα που Ορίστηκαν/Χρησιμοποιούνται) στον Server.

Ο Server λαμβάνει το αιτημα αυτό, στο οποίο εμπεριέχονται το Auth Token της συσκευής, ο αριθμός προφίλ συσκευής και το PIN του χρήστη. Αφότου γίνει επαλήθευση των στοιχείων αυτών, παράγεται τυχαία ένα αλφαριθμητικό 30 χαρακτήρων γνωστό ως Wear Token, αποθηκεύεται hashed στο προφίλ της συσκευής, στην ΒΔ, και αποστέλλεται στην συσκευή Android. Αφότου παραληφθεί από την συσκευή Android γίνεται αποστολή του Wear Token στην συσκευή Android Wear. Πλέον μπορούν να γίνουν αιτήματα ξεκλειδώματος μέσω Android Wear.

Όταν αιτείται ξεκλείδωμα μέσω Android Wear, γίνεται αποστολή του Wear Token (που βρίσκεται στο Smartwatch) και του Auth Token (που βρίσκεται στο Smartphone) στον Server και γίνεται ταυτοποίησή τους. Αν αντιστοιχούν σε υπάρχον προφίλ συσκευής, γίνεται ενεργοποίηση του μηχανισμού ξεκλειδώματος.

Εάν κατα κάποιο σημείο αποτύχει κάποια επαλήθευση, γίνεται αποστολή μηνύματος αποτυχίας στην συσκευή Android.

#### 8.4.4 Αλλαγή PIN χρηστών

Όπως αναφέρθηκε προηγουμένως (2.2 Σύντομη Περιγραφή Λογισμικού Πελάτη - Pi-Lock Client), μία από τις λειτουργίες της εφαρμογής πελάτη για Android είναι η δυνατότητα αλλαγής του προσωπικού PIN του χρήστη, εφόσον το επιθυμεί. Για να γίνει αυτό ακολουθείται μια παρόμοια διαδικασία με αυτή του δεύτερου σταδίου εξουσιοδότησης πρόσβασης.

Ο χρήστης υποχρεούται να πληκτρολογήσει το παλιό του PIN, μαζί με το νέο. Μόλις γίνει η πληκτρολόγηση, αποστέλλεται στον Server ένα αίτημα άλλαγής PIN με το παλιό και το νέο PIN, το Auth Token και το νούμερο προφίλ συσκευής. Ο Server τα επιβεβαιώνει, και εφόσον είναι επιτυχής η επιβεβαίωση, γίνεται άλλαγή του PIN στο νέο, που επιθυμεί ο χρήστης. Εαν τα στοιχεία που δώθηκαν δεν είναι έγκυρα, γίνεται απόρριψη του αιτήματος.

#### 8.4.5 Ασφάλεια

Το σύστημα που αναλύθηκε παραπάνω έχει δημιουργηθεί με την ασφάλεια κατά nou. Για αυτόν τον λόγο δεν επιτρέπεται να γίνει αίτημα ξεκλειδώματος από μή εξουσιοδοτημένες συσκευές. Πρέπει υποχρεωτικά μία συσκευή να περάσει από το πρώτο στάδιο πριν να αποπειραθεί να κάνει αίτημα ξεκλειδώματος.

Αν ένας χρήστης με κακόβουλες προθέσεις επιλέξει να παραλήψει το πρώτο στάδιο, για να μπορέσει να πραγματοποιήσει ξεκλειδωματα όταν πρέπει πρωτίστος να καταφέρει να "μαντέψει" το Auth Token, είτε μαντεύοντας το Auth Token, είτε δοκιμάζοντας όλους τους δυνατούς συνδυασμούς χαρακτήρων προκειμένου να βρεθεί το σωστό Auth Token. Και στις 2 περιπτώσεις, είναι εξαιρετικά δύσκολο να βρεθεί το Auth Token. Στην περίπτωση που ο χρήστης επιλέξει να μαντέψει το Auth Token, εκμεταλλευόμενος την γεννήτρια τυχαιότητας, είναι αδύνατον να μαντέψει το αποτέλεσμα της συνάρτησης σε μία συγκεκριμένη στιγμή, καθώς χρησιμοποιείται η βέλτιστη πυγή τυχαιότητας του λειτουργικού συστήματος την συγκεκριμένη στιγμή μέσω της κλάσης `SystemRandom` της Python, η οποία εγγυάται ασφαλή τυχαιότητα[38].

Στην περίπτωση που ο κακόβουλος χρήστης επιλέξει να μαντέψει το Auth Token εξαντλώντας όλους τους πιθανούς συνδυασμούς χαρακτήρων, λόγω του μεγέθους και της πολυπλοκότητας του Auth Token ( $50^{32}$  χαρακτήρες, μικρά γράμματα, σύμβολα), θα πρέπει να δοκιμάσει κατά μέσο όρο  $50^{50}/2 = \sim 4,44089209810^{84}$  πιθανούς συνδυασμούς χαρακτήρων (με επαναλήψεις). Παρομοίως, για το Wear Token, θα χρειαστεί να εκτελέσει, κατά μέσο όρο  $50^{32}/2 = \sim 1,16415321810^{54}$  πιθανούς συνδυασμούς χαρακτήρων (με επαναλήψεις). Αυτό καθιστά πολύ δύσκολο να σπάσει κάποιο από τα δύο tokens, με αποτέλεσμα να είναι σχεδόν αδύνατον να εισβάλλει κάποιος στο σύστημα. Σε αυτό το σημείο αξίζει να προστεθεί οτι προκειμένου ο εξυπηρετητής να εγκρίνει το αίτημα ξεκλειδώματος χρειάζεται να μαντέψει ο κακόβουλος χρήστης, παράλληλα με το Auth Token, και το PIN του χρήστη. Ο εξυπηρετητής δεν διαφοροποιεί τις απαντήσεις του προς την εφαρμογή πελάτη αν είναι σωστό ένα εκ των δύο στοιχείων πρόσβασης, δηλαδή δεν θα δώσει κάποιο στοιχείο ως προς το τι είναι λανθασμένο. Πρέπει να είναι και τα δύο στοιχεία σωστά προκειμένου να εγκριθεί το αίτημα και στην συγκεκριμένη περίπτωση, το να μαντέψει κάποιος το Auth Token **παράλληλα** με το PIN του χρήστη είναι εξαιρετικά δύσκολο.

Τέλος, η διεξαγωγή της μεταφοράς δεδομένων από τον Server προς την εφαρμογή πελάτη γίνεται με κρυπτογράφηση SSL και όπως θα αναφερθεί αργότερα, στο σχετικό κεφάλαιο, γίνεται κλείδωμα του πιστοποιητικού που χρησιμοποιείται στην εφαρμογή πελάτη, προκειμένου να είναι αδύνατον να γίνει παραχάραξη του. Προκειμένου, επίσης, να αποτραπούν επιθέσεις τύπου Protocol Downgrading, χρησιμοποιείται HSTS (HTTP Strict Transfer Security)[25].

## 8.5 Λειτουργία Heartbeat

Προκειμένου να καταστεί δυνατό, με κάποιο τρόπο να γίνεται αντιληπτό αν ο Server είναι ενεργός, έπρεπε να υλοποιηθεί ένας μηχανισμός αναφοράς κατάστασης. Αυτός ο μηχανισμός είναι γνωστός ως Heartbeat, στην περίπτωση του PiLock. Αποτελείται από μία σελίδα στην οποία, όπως και στο υπόλοιπο API, χρησιμοποιείται JSON, προκειμένου να αναφέρει την έκδοση του Server, καθώς επίσης και την κατάσταση του Server (Figure 8.3).

To Heartbeat χρησιμοποιείται προκειμένου να μπορέσει να αντιληφθεί η εφαρμογή Android αν το σύστημα είναι ενεργό, πριν να αποστείλλει κάποιο αίτημα ξεκλειδώματος. Χρησιμοποιείται επίσης και για να λειτουργήσει ο γενικός διακόπτης, όπως θα αναλυθεί παρακάτω. Η έκδοση του λογισμικού του Server μεταδίδεται προκειμένου να γνωρίζει η εφαρμογή πελάτη αν είναι ενημερωμένη ώστε να μπορεί να λειτουργήσει με την παρούσα έκδοση λογισμικού στον Server.

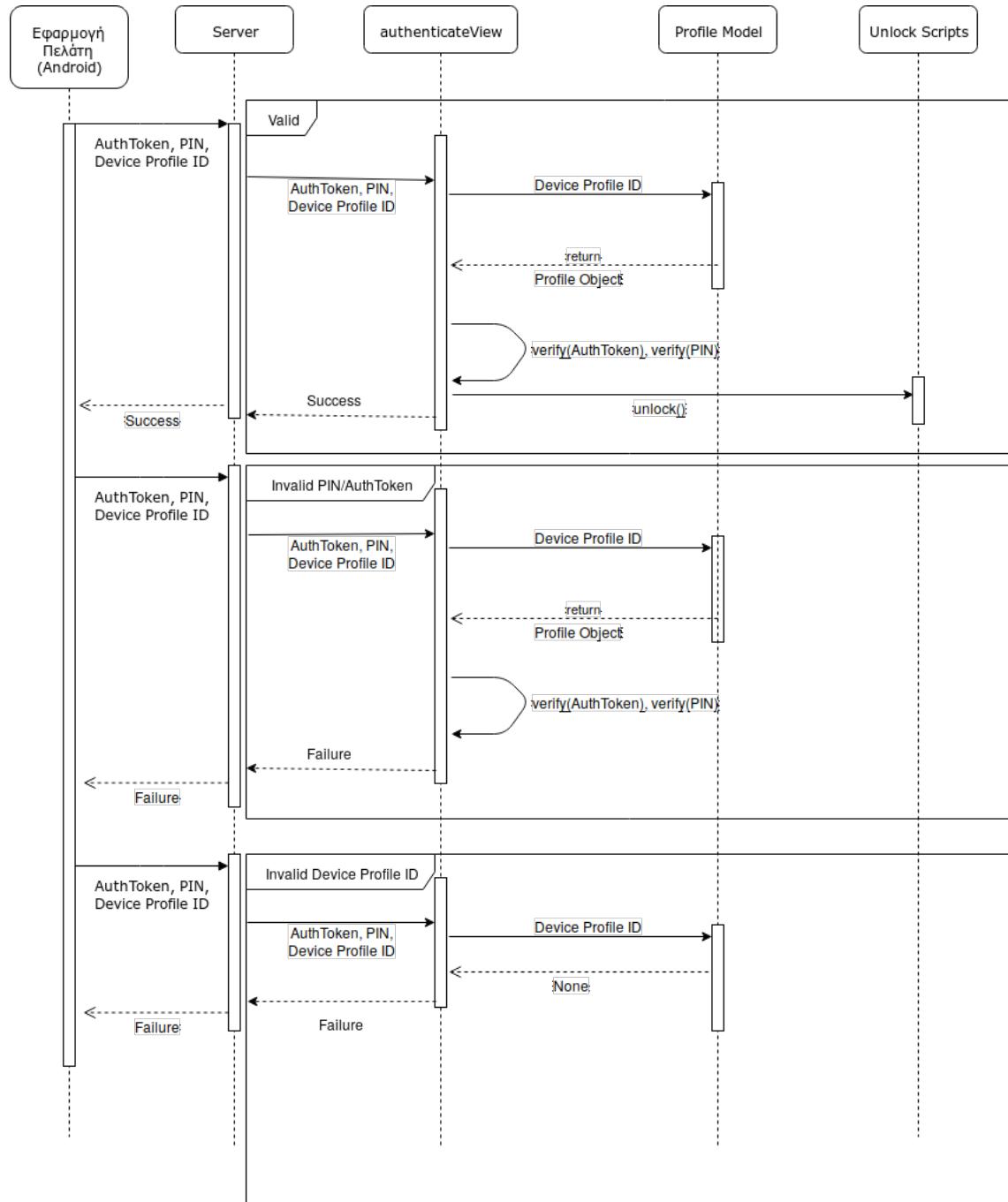
## 8.6 Γενικός Διακόπτης

Πολλές φορές καθίσταται αναγκαίο να υπάρχει ένας γενικός διακόπτης, προκειμένου να αποκλείει αυτόμια απόπειρα ξεκλειδώματος ή πρόσβασης στο σύστημα του PiLock. Αυτό καθίσταται ιδιαίτερα χρήσιμο σε περιπτώσεις που έχει συμβεί κάποιο εξαιρετικά σημαντικό συμβάν και να χρειάζεται να διακοπεί η οποιαδήποτε πρόσβαση σε οποιονδήποτε χρήστη.

Για να υλοποιηθεί αυτό, δημιουργήθηκε ένα αρχείο διαμόρφωσης σε γλώσσα σήμιανσης YAML, γνωστό ως config.yml το οποίο τοποθετείται στον αρχικό κατάλογο του Project κατά την εγκατάσταση. Μέσα σε αυτό υπάρχει η εξής μεταβλητή:

<sup>1</sup> enabled : True

Εάν η μεταβλητή αλλάζει από τον χρήστη σε False, αυτόματα πλέον το μήνυμα που μεταδίδεται κατά την διαδικασία του Heartbeat αλλάζει. Συγκεκριμένα, πλέον η κατάσταση που μεταδίδει το Heartbeat αλλάζει σε status="LOCKED". Αυτό σηματοδοτεί την εφαρμογή πελάτη να ακυρώσει την διαδικασία που προτάθηκε από τον χρήστη, εφόνον λάβει αυτό το μήνυμα.



Εικόνα 8.2: Στάδιο Ξεκλειδώματος

JSON Raw Data Headers

Save Copy Pretty Print

```
{  
  "status": "ALIVE",  
  "version": "0.3.1"  
}
```

Εικόνα 8.3: Μήνυμα Heartbeat

# Κεφάλαιο 9

## Η εφαρμογή Android - PiLock Client

Η εφαρμογή πελάτη, γνωστή ως PiLock Client είναι μια Smartphone εφαρμογή (Smartphone App), κατασκευασμένη για Android Smartphones. Μπορεί να λειτουργήσει σε οποιαδήποτε συσκευή που διαθέτει έκδοση Android Kitkat (API Level 19) ή ανώτερη. Σύμφωνα με την διανομή αγοράς εκδόσεων Android<sup>[?]</sup> για την συγκεκριμένη έκδοση Android, καλύπτεται το 90% των συσκευών που βρίσκονται αυτή την στιγμή στην αγορά, οπότε είναι ιδανική η συγκεκριμένη επιλογή έκδοσης. Η εφαρμογή Android είναι εξ'ολοκλήρου κατασκευασμένη στο Android Studio (βλ. 5.4 Προγραμματιστικό Περιβάλλον), και χρησιμοποιεί για λογισμικό διαχείρησης εκδόσεων, όπως και το λογισμικό του εξυπηρετητή, το Git.

### 9.1 Activities

Η εφαρμογή αποτελείται από διάφορα Activities. Ός Android Activity ορίζεται μία μοναδική, στοχευμένη γραφική διεπαφή χρήστη από την οποία ο χρήστης μπορεί να συγκεντρώνεται σε μία συγκεκριμένη κατάσταση<sup>[28]</sup>. Στην τελευταία έκδοση του PiLock (0.3.1), υπάρχουν συνολικά τέσσερα (4) Activities:

- **LoginActivity:** Το Activity υπεύθυνο για την εισαγωγή στοιχείων και σύνδεση του χρήστη στο σύστημα του PiLock. Μέσω αυτού υλοποιείται το 1ο στάδιο εξουσιοδότησης. Αποτελείται από 2 πεδία κειμένου που δέχονται το Username και το Password του χρήστη καθώς επίσης και ένα πλήκτρο, μέσω του οποίου γίνεται αποστολή των στοιχείων αυτών στον Server.
- **PINEntryActivity:** Activity, μέσω του οποίου ο χρήστης εισάγει το PIN του και στέλνει αιτήματα ξεκλειδώματος. Μέσω αυτού υλοποιείται το 2ο στάδιο εξουσιοδότησης. Αποτελείται από ένα πεδίο κειμένου στο οποίο ο χρήστης μπορεί να εισάγει το PIN του καθώς επίσης και ένα πλήκτρο με το οποίο γίνεται αποστολή των στοιχείων αυτών στον Server.
- **ChangePinActivity:** Υπεύθυνο για την αλλαγή PIN του χρήστη. Αποτελείται από 3 πεδία κειμένου στα οποία ο χρήστης μπορεί να εισάγει το παλιό του PIN, το νέο του PIN καθώς επίσης και το νέο του PIN για δεύτερη φορά για επιβεβαίωση. Τέλος, αποτελείται και από ένα πλήκτρο με το οποίο γίνεται αποστολή του αιτήματος αλλαγής PIN στον Server.

- **SettingsActivity:** Μαζί με το SettingsFragment, αποτελούν την ουθόνη ”ρυθμίσεων” της εφαρμογής. Μέχρι και την τελευταία έκδοση της εφαρμογής, η ουθόνη ρυθμίσεων χρησιμοποιείται προκειμένου να οριστεί η διεύθυνση του Server, κατά την πρώτη εκτέλεση της εφαρμογής. Σε μία επόμενη έκδοση της εφαρμογής, θα προστεθεί επιλογή να ορίζεται ο Server ως προσβάσιμος απολειτικά μέσω WiFi (βλ. 14 Μελλοντική Επέκταση του PiLock).

## 9.2 Αιτήματα HTTPS

Όπως αναφέρθηκε προηγουμένως, όλες οι ανταλλαγές πληροφοριών ανάμεσα στην εφαρμογή και τον Server γίνονται χρησιμοποιόντας το πρωτόκολλο HTTPS. Προκειμένου να καταστεί εύχρηστο, δημιουργήθηκε μια βιβλιοθήκη αποτελούμενη από 5 κλάσεις και ένα enum, η οποία αναλαμβάνει την αποστολή και λήψη πληροφοριών προς και από τον Server.

Η κύρια κλάση-γονέας (Superclass) λέγεται `HttpsRequest`, και χρησιμοποιείται προκειμένου να κρατά τις πληροφορίες ενός οποιουδήποτε HTTPS αιτήματος, όπως για παράδειγμα την διεύθυνση του αποδοχέα, τις παραμέτρους, καθώς επίσης και τις πληροφορίες που λαμβάνονται αφότου εκτελεστεί το αίτημα, όπως τον κωδικό απάντησης, καθώς επίσης και την ίδια την απάντηση. Μέσα σε αυτή την κλάση ορίστηκε η εσωτερική κλάση `RequestNotExecutedException`, η οποία αποτελεί κλάση εξαίρεσης και χρησιμοποιείται σε περίπτωση που ο χρήστης ζητήσει να λάβει τα δεδομένα της απάντησης πρωτού εκτελεστεί το αίτημα. Τέλος, ορίζεται και το `HttpsRequestListener` interface, το οποίο καλείται όταν ολοκληρωθεί ένα αίτημα. Αυτό καθιστά εύκολη την άμεση αυτοματοποίηση διαδικασιών με την λήξη των αιτημάτων. Η κλάση `HttpsRequest` κληρονομείται από δύο κλάσεις γνωστές ως `HttpsGET` και `HttpsPOST`, υπεύθυνες για αποστολή αιτημάτων GET και POST, αντίστοιχα.

Η διαδικασία αποστολής των αιτημάτων αυτών είναι η εξής:

1. Δημιουργία αντικειμένου τύπου `HttpsGET` ή `HttpsPOST`. Στον κατασκευαστή (Constructor), δίνονται ως ορίσματα η διεύθυνση (URL) του παραλήπτη και (προαιρετικά) οι παράμετροι του αιτήματος, σε μορφή `HashMap<String, String>`. Μέσω της κλάσης `QueryBuilder`, γίνεται κτήσιμο του σώματος του αιτήματος, από τις δοσμένες παραμέτρους.
2. Σε περίπτωση που χρειαστεί να εκτελεστεί κάποια ενέργεια με την λήξη του αιτήματος, πρέπει να γίνει ορισμός του Request Listener μέσω της μεθόδου `setRequestListener(HttpsRequest.HttpsRequestListener listener)`. Στο σώμα του `HttpRequestListener`, θα πρέπει να γίνει υλοποίηση της μεθόδου `onRequestCompleted()`. Μόλις λήξει το αίτημα, και κλείσει η σύνδεση, θα εκτελεστεί αυτή η μέθοδος.
3. Κλήση της μεθόδου `SendGET(Context c)` ή `SendPOST(Context c)`, ανάλογα τον τύπο του αιτήματος.
4. Μετά την ολοκλήρωση του αιτήματος, μπορούμε να πραγματοποιηθεί ανάκτηση του κωδικού απόκρισης καθώς επίσης και την ίδια την απόκριση καλόντας την μέθοδο `getResponseBody()` και `getResponse()`. Είναι καλό να γίνεται έλεγχος του αιτήματος για σφάλμα χρησιμοποιόντας την μέθοδο `hasError()`, ανάκτηση του σε περίπτωση που υπάρχει μέσω της μεθόδου `getError()` και να γίνεται

σύγκριση με τις τιμές του `HttpsConnectionError` για να διαπιστωθεί ποιο είναι το σφάλμα. Να σημειωθεί ότι το enum `HttpsConnectionError` συμβολίζει μόνο σφάλματα σύνδεσης, και όχι σφάλματα μέσα στην ίδια την απόκριση. Για να γίνει διάγνωση της ίδιας της απόκρισης καλό είναι να γίνεται ανάκτηση και σύγκριση του κωδικού απάντησης με τους ήδη γνωστούς κωδικούς απάντησης του πρωτόκολλου HTTP (HTTP Response Codes).

5. Αφότου γίνει ανάκτηση της απόκρισης, μπορεί να αναλυθεί μέσω της βιβλιοθήκης αποκωδικοποίησης JSON της Java.

### 9.2.1 Το πρόβλημα με τα Αυτο-Υπογεγραμμένα Πιστοποιητικά SSL

Ένα από τα δυσκολότερα προβλήματα που συναντήθηκαν κατά την πρώτη φάση υλοποίησης του PiLock ήταν η χρήση πιστοποιητικών SSL. Πολλές φορές, οι εγκαταστάσεις του PiLock θα χρησιμοποιούνται σε τοπικό επίπεδο (προσβάσιμες αποκλειστικά μέσω WiFi), πράγμα το οποίο καθιστά την απόκτηση πιστοποιητικού από μία εγκεκριμένη αρχή πιστοποίησης δαπανηρή και πολλές φορές αδύνατη. Για αυτό τον λόγο χρησιμοποιούνται αυτο-υπογεγραμμένα πιστοποιητικά SSL (Self Signed Certificates). Λόγο της δομής του, το σύστημα Android εμπιστεύεται πιστοποιητικά μόνο από έγκυρες αρχές πιστοποίησης, αλλά μπορεί να ρυθμιστεί, εφόσον το επιθυμεί ο προγραμματιστής, να εμπιστεύεται πιστοποιητικά από τρίτους.

Ο τρόπος ο οποίος επινοήθηκε προκειμένου να μπορεί να γίνει χρήση αυτο-υπογεγραμμένου πιστοποιητικού είναι ο εξής: Αφότου ο χρήστης δημιουργήσει το πιστοποιητικό για την εγκατάσταση του PiLock και το εγκαταστήσει στον Server, θα πρέπει να γίνει αντιγραφή του πιστοποιητικού (όχι του μυστικού κλειδιού) στον αρχικό κατάλογο στον χώρο αποθήκευσης του SmartPhone, σε ένα αρχείο με όνομα `pilock.crt`, και σε μορφή PEM. Έπειτα, κατά την αποστολή αιτημάτων HTTPS, γίνεται ανάκτηση του συγκεκριμένου αρχείου από την κλάση `CustomSSLTruster` και ορίζεται ως άξιο εμπιστοσύνης στο αίτημα. Για να γίνει ανάκτηση του πιστοποιητικού από τον χώρο αποθήκευσης του Smartphone χρειάζεται να δωθεί η άδεια πρόσβασης στον χώρο αποθήκευσης από τον χρήστη, κατά την διάρκεια εκτέλεσης του PiLock.

Αυτό, εκτός από το να βοηθά στην λειτουργία του συστήματος, καθιστά ασφαλή την αποστολή αιτημάτων αποφεύγοντας επιθέσεις προσωποποίησης (Impersonation Attack), καθώς γίνεται σύγκριση του πιστοποιητικού του Server με αυτό που έχει αποθέσει ο χρήστης στην κάρτα μνήμης, και η σύνδεση απορρίπτεται αν δεν ταιριάζουν τα πιστοποιητικά.

## 9.3 Μηχανισμός Heartbeat

Όπως αναφέρθηκε προηγουμένως (βλ. 8.5 Λειτουργία Heartbeat), έχει υλοποιηθεί στον Server μηχανισμός Heartbeat, υπεύθυνος για την αναφορά της κατάστασης του Server καθώς επίσης και για την επιβεβαίωση ενημερωμένης εφαρμογής. Ο μηχανισμός Heartbeat έχει υλοποιηθεί στην εφαρμογή πελάτη με την ομόνυμη κλάση `Heartbeat`, η οποία διαθέτει όλες τις αναγκαίες μεθόδους για την λήψη της κατάστασης του Server, καθώς επίσης και ένα `listener interface` το οποίο είναι υπεύθυνο για εκτέλεση μεθόδων ανάλογα με το αποτέλεσμα του Heartbeat.

To Heartbeat εκτελείται πριν να γίνει το οποιοδήποτε αίτημα προς τον Server και, αν ο Server δεν βρεθεί ή αν είναι κλειδωμένος, σταματά την διαδικασία του αιτήματος και δεν αποστέλλεται στον Server. Πιο συγκεκριμένα:

1. Γίνεται ανάγνωση του εσωτερικού χώρου αποθήκευσης του κινητού για την ύπαρξη αντιγράφου του πιστοποιητικού SSL που χρησιμοποιεί ο Server. Αν δεν βρεθεί, το Heartbeat χαρακτηρίζεται ως αποτυχημένο.
2. Στέλνεται αίτημα GET χωρίς παραμέτρους στην ρίζα (Root) της τοποθεσίας του PiLock Server. Αν δεν υπάρχει απάντηση, ή αν υπάρχει κάποιο πρόβλημα κατά την επικοινωνία, το Heartbeat χαρακτηρίζεται ως αποτυχημένο.
3. Γίνεται ανάλυση της απόκρισης του Server. Για αρχή γίνεται έλεγχος του πεδίου κατάστασης (*status*). Αν ο Server αναφέρει οτι είναι κλειδωμένος (*LOCKED*), το Heartbeat χαρακτηρίζεται ως αποτυχημένο και εμφανίζεται μήνυμα κλειδωμένου Server στον χρήστη.
4. Αν ο Server αναφέρει οτι είναι "ζωντανός" (*ALIVE*), ακολουθεί σύγκριση του πεδίου έκδοσης λογισμικού του Server, με μία προκαθορισμένη τιμή στον κώδικα της εφαρμογής (μεταβλητή *LatestServerVersion*, κλάση *VersionChecker*), που αναφέρει με ποια έκδοση Server είναι συμβατή η παρούσα έκδοση της εφαρμογής. Αν διαπιστώθει οτι διαφέρουν (μέθοδος *CheckVersion*(*VersionReportedByServer*), κλάση *VersionChecker*) το Heartbeat χαρακτηρίζεται ως αποτυχημένο και εμφανίζεται μήνυμα μη ενημερωμένης εφαρμογής στον χρήστη.
5. Αν δεν έχει μέχρι τώρα χαρακτηριστεί το Heartbeat ως αποτυχημένο, χαρακτηρίζεται ως επιτυχημένο και ολοκληρώνεται η εργασία.

Παρομοίως με τα αιτήματα HTTPS, ο χρήστης πρέπει να ορίσει έναν listener για το Heartbeat. Συγκεκριμένα, πρέπει να υλοποιήσει 3 μεθόδους:

- **onHeartbeatFinished()**: Εκτελείται με την λήξη του Heartbeat, ανεξάρτητα με το αποτέλεσμα.
- **onHeartbeatSuccess()**: Εκτελείται έπειτα από την **onHeartbeatFinished**, αν το Heartbeat ήταν επιτυχές.
- **onHeartbeatFailure()**: Εκτελείται έπειτα από την **onHeartbeatFinished**, αν το Heartbeat ήταν αποτυχημένο.

## 9.4 Κρυπτογράφηση Τεκμηρίων Πρόσβασης

Όπως αναφέρθηκε προηγουμένως, κατά το πρώτο στάδιο αυθεντικοποίησης μίας συσκευής με τον Server του PiLock, γίνεται αποστολή ενός τεκμηρίου πρόσβασης, γνωστό ως Authorization Token (Auth Token). Αυτό το τεκμήριο, στην περίπτωση που υποκλαπεί, η διαδικασία αυθεντικοποίησης θεωρείται μη ασφαλής. Υποκλοπή μπορεί να συμβεί σε περίπτωση που η εφαρμογή εγκαταστηθεί σε Rooted συσκευή Android. Σε τέτοιου είδους συσκευές, μία εφαρμογή μπορεί να αποκτήσει αυξημένα προνόμια πρόσβασης και να έχει πρόσβαση σε διάφορα στοιχεία, άλλων εφαρμογών ή του συστήματος, που κανονικά δεν θα είχε<sup>[29]</sup> (σε μια συσκευή χωρίς Root πρόσβαση). Εφόσον το τεκμήριο πρόσβασης αποθηκεύεται στα Shared Preferences της συσκευής,

μια κακόβουλη εφαρμογή, σε περίπτωση που η συσκευή είναι Rooted, μπορεί να έχει πρόσβαση στα Shared Preferences άλλων εφαρμογών κατά βούληση.

Προκειμένου να καταστεί αδύνατον να γίνει ανάκτηση του τεκμήριου πρόσβασης από Rooted συσκευές, χρησιμοποιείται το σύστημα Android Keystore<sup>[30]</sup>. Για να γίνει η υλοποίησή του, δημιουργήθηκε η κλάση `KeystoreHelper`, σκοπός της οποίας είναι να κρυπτογραφεί και να αποκρυπτογραφεί κάποιο δωσμένο αλφαριθμητικό μέσω του Android Keystore, χρησιμοποιώντας τον αλγόριθμο RSA. Αφότου δημιουργηθεί ένα αντικείμενο της κλάσης, οι μέθοδοι `Encrypt(StringToEnc)` και `Decrypt(Ciphertext)`, χρησιμοποιούνται για να γίνει κρυπτογράφηση ενός αλφαριθμητικού και αποκρυπτογράφηση αντίστοιχα. Συγκεκριμένα, γίνεται κρυπτογράφηση του Auth Token, και έπειτα γίνεται αποθήκευση του κρυπτογραφημένου Token στα Shared Preferences της συσκευής. Με αυτό τον τρόπο, καθίσταται ασφαλής η αποθήκευση ευαίσθητων δεδομένων της εφαρμογής.

## Κεφάλαιο 10

# Η εφαρμογή Android Wear

Με την κυκλοφορία της έκδοσης 0.3.0, έγινε δημιουργία μιας συνοδευτικής εφαρμογής για χρήση με Android Wear Smartwatches. Εφόσον ο χρήστης συγχρονίσει την εφαρμογή πελάτη με την εφαρμογή Android Wear, μπορεί να εκτελέσει ξεκλειδώματα μέσω αυτής, χωρίς να χρειαστεί να εισάγει το PIN του.

Προκειμένου να γίνει η μεταφορά του Wear Token (βλ. 8.4.3 Ξεκλειδωματα μέσω Android Wear - Wear Unlock), γίνεται χρήση του MessageClient API του Android Wear<sup>[31]</sup>. Αρχικά, με την εκκίνηση της εφαρμογής, γίνεται εντοπισμός συνδεδεμένης με το Smartphone συσκευής Android Wear. Μόλις γίνει σύνδεση, μπορεί να γίνει συγχρονισμός μέσω του MessageClient API και του Server. Αφότου γίνει η αυθεντικοποίηση και η ανάκτηση του Wear Token από τον Server, γίνεται αποστολή του Wear Token που έχει ληφθεί από τον Server, στην συσκευή Android Wear, όπου και γίνεται αποθήκευσή του.

Αντίστοιχα, μόλις ο χρήστης θελήσει να πραγματοποιήσει ξεκλειδωματα μέσω του Smartwatch του, το μόνο που οφείλει να κάνει είναι να εκκινήσει την εφαρμογή από το Smartwatch και να αγγίξει το πλήκτρο ξεκλειδώματος, αναπαριστόμενο από το ξεκλειδωτό λουκέτο. Με το άγγιγμα στο πλήκτρο ξεκλειδώματος γίνεται αποστολή του Wear Token μέσω του MessageClient API στην συσκευή Android με την οποία είναι συνδεδεμένο το Smartwatch. Μόλις ληφθεί από την εφαρμογή πελάτη, γίνεται εκκίνηση του PINEntryActivity και στέλνεται το Wear Token μαζί με το Auth Token και το Device Profile ID στον Server. Εφόσον η επαλήθευση είναι επιτυχής, γίνεται ξεκλειδωματα.

# Κεφάλαιο 11

## Ο Πίνακας Διαχείρησης του PiLock

Στην πρώτη δημόσια έκδοση του PiLock (0.2.0), έγινε η προσθήκη του πίνακα διαχείρησης του PiLock, γνωστού ως AdminCP (Administration Control Panel). Πριν να γίνει η προσθήκη αυτού, οι χρήστες υποχρεούνταν να χρησιμοποιούν τον ενσωματωμένου πίνακα διαχείρησης που παρέχει το Django, για την διαχείρηση των χρηστών και των μοντέλων του συστήματος.

Στον πυρήνα του, ο πίνακας διαχείρησης, προσβάσιμος στο [https://<PiLock\\_Root>/AdminCP](https://<PiLock_Root>/AdminCP), όπου PiLock\_Root η διεύθυνση της εγκατάστασης του PiLock, περιέχει τις πιο αναγκαίες λειτουργίες για την διαχείρηση της πλατφόρμας. Αυτές είναι:

- Προσθήκη, διαγραφή χρηστών. Οι χρήστες μπορούν να οριστούν ως λειτουργικό προσωπικό (Staff), προκειμένου να μπορούν να έχουν πρόσβαση στον πίνακα διαχείρησης. Όσοι χρήστες δεν έχουν οριστεί ως Staff, δεν μπορούν να έχουν πρόσβαση στον πίνακα διαχείρησης του PiLock, αλλά μπορούν να πραγματοποιήσουν ξεκλείδωμα.
- Διαγραφή υπάρχοντων προφίλ συσκευών. Σε περίπτωση που κάποιος χρήστης επιθυμεί, μπορεί να ζητήσει από τον διαχειριστή του συστήματος να διαγράψει το προφίλ της συσκευής του, προκειμένου να ξανακάνει σύνδεση στο σύστημα του PiLock, σε περίπτωση που χαθεί η συσκευή του ή για κάποιο λόγο διαγραφεί το τεκμήριο πρόσβασης από την συσκευή του.
- Προβολή του ιστορικού πρόσβασης. Θα αναλυθεί εκτενώς παρακάτω.
- Πραγματοποίηση ξεκλειδώματος απευθείας από τον πίνακα διαχείρησης. Κάποιες φορές, είναι επιθυμητό να γίνεται ξεκλείδωμα απευθείας από τον πίνακα διαχείρησης του PiLock (για λόγους ευελιξίας, ή σε περίπτωση που κάποιος διαχειριστής δεν έχει εγκατεστημένη την εφαρμογή στο κινητό του), για αυτό τον λόγο έγινε προσθήκη του μηχανισμού αυτού.
- Σύστημα ειδοποιήσεων. Προβάλλει ειδοποιήσεις σχετικές με την υγεία του συστήματος. Θα αναλυθεί εκτενώς παρακάτω.

## 11.1 Ημερολόγιο Πρόσβασης - Access Log

Κάποιες φορές, ειδικά σε περιπτώσεις που το PiLock χρησιμοποιείται από πολλούς χρήστες, ή σε περίπτωση που χρησιμοποιηθεί σε κάποιο εταιρικό περιβάλλον, είναι θεμιτό να κρατείται αρχείο με το ιστορικό οποιασδήποτε πρόσβασης στο σύστημα, για λόγους ασφαλείας. Στην έκδοση 0.2.0, δημιουργήθηκε ένα τέτοιο σύστημα. Συγκεκριμένα, χρησιμοποιόντας το `AccessAttempt` μοντέλο (βλ. 8.3 Μοντέλα που Ορίστηκαν/Χρησιμοποιούνται), κάθε φορά που γίνεται μια απόπειρα σύνδεσης ή ξεκλειδώματος, δημιουργείται ένα νέο αντικείμενο του `AccessAttempt`, και δίνονται σε αυτό κάποια από τα δεδομένα σχετικά με τον χρήστη που πραγματοποίησε την απόπειρα αυτή, δηλαδή το όνομα χρήστη του, η διεύθυνση IP του, εάν είναι απόπειρα ξεκλειδώματος, εάν η απόπειρα είναι επιτυχής, καθώς επίσης και η ημερομηνία και η ώρα της απόπειρας.

Τα στοιχεία αυτά μπορούν να χρησιμοποιηθούν από τον διαχειριστή της εγκατάστασης προκειμένου να διασταυρώθουν με κάποιο γεγονός σχετικό με την ασφάλεια του κτιρίου της εγκατάστασης (σε περίπτωση που γίνει κάποια επίθεση/ληστεία στο κτήριο από κάποιον κακόβουλο εξουσιοδοτημένο χρήστη), ή για να εξιχνιαστούν επιθέσεις επανειλημμένων αποπειρών ξεκλειδώματος ή/και επιθέσεων άρνησης υπηρεσίας (Denial of Service Attacks).

## 11.2 Σύστημα ειδοποιήσεων

Στην έκδοση 0.3.0 έγινε προσθήκη στον πίνακα διαχείρησης ενός συστήματος ειδοποίησεων. Οι ειδοποιήσεις αυτές στόχο έχουν να ειδοποιούν τον χρήστη όποτε υπάρχει κάποιο συμβάν που μπορεί να κλονίσει την ασφάλεια του συστήματος. Υπάρχουν (μέχρι την έκδοση 0.3.1) τρία είδη ειδοποιήσεων:

- **Debug Mode:** Ενεργοποιείται όταν το PiLock τρέχει με ενεργοποιημένη την λειτουργία Debug. Κατά την λειτουργία αυτή, δεν πραγματοποιούνται ξεκλειδώματα, και το σύστημα χρησιμοποιείται μόνο για αποσφαλμάτωση (κατά την ανάπτυξη νέων λειτουργιών ή την διόρθωση ήδη υπάρχοντων). Χαρακτηριστικό αυτής της λειτουργίας είναι η εμφάνιση όλων των μεταβλητών κατά την στιγμή εκτέλεσης σε περίπτωση που υπάρξει κάποιο σφάλμα. Για αυτό τον λόγο αυτόματα απενεργοποιείται ο μηχανισμός ξεκλειδωμάτων ενόσω η λειτουργία αποσφαλμάτωσης είναι ενεργή, καθώς μπορεί να γίνει διαρροή ευαίσθητων δεδομένων μέσω αυτής και να επιτραπεί πρόσβαση σε μη εξουσιοδοτημένο/κακόβουλο χρήστη.
- **Update:** Ενεργοποιείται όποτε υπάρχει διαθέσιμη κάποια ενημέρωση για το PiLock. Ο έλεγχος γίνεται με χρήση ενός Cron Job, προγραμματισμένου να εκτελείται κάθε 60 λεπτά, χρησιμοποιόντας το Crontab του συστήματος, καθώς επίσης και το `django-cron`<sup>1</sup> module προκειμένου να γίνει εύκολη ενσωμάτωση με το λογισμικό του Server. Ο έλεγχος γίνεται συγκρίνοντας το commit hash του Server που υπάρχει εγκαταστημένος, με το τελευταίο commit hash που υπάρχει στο master branch στο αποθετήριο του λογισμικού στο GitHub, καλώντας το API του GitHub.

<sup>1</sup><https://github.com/Tivix/django-cron>

- **Security:** Δεν έχει υλοποιηθεί ακόμα. Θα χρησιμοποιείται προκειμένου να ειδοποιήσει για κάποιο ύψη ασφαλείας σχετικό με την πλατφόρμα, για παράδειγμα όταν κάποιος χρήστης πραγματοποιεί συνεχόμενα αποτυχημένες απόπειρες ξεκλειδώματος, ή αν είναι αποσυνδεδεμένο το Arduino από το Raspberry Pi.

Οι ειδοποιήσεις εμφανίζονται στην κεντρική σελίδα του πίνακα διαχείρησης, με την μορφή Bootstrap Alerts.

# Κεφάλαιο 12

## Εγκατάσταση και ρύθμιση του PiLock Server

Ένα εκ των σημαντικότερων σημείων κατά τον σχεδιασμό του PiLock ήταν ο σχεδιασμός ενός συστήματος αυτοματοποιημένης εγκατάστασης, προκειμένου να μην χρειάζεται ο τελικός χρήστης να δαπανήσει χρόνο προκειμένου να ρυθμίσει το λογισμικό. Αυτό γίνεται μέσω δύο σεναρίων γραμμένων σε Shell, που στόχος τους είναι να εκτελέσουν σχεδόν όλα τα απαιτούμενα βήματα προκειμένου να εγκαταστηθεί και να ρυθμιστεί επιτυχώς το PiLock.

Το πρώτο σενάριο, γνωστό ως `setup_apache.sh` είναι υπεύθυνο για την εγκατάσταση του Webserver υπεύθυνου για την εκτέλεση του PiLock καθώς επίσης και για την αρχική ρύθμιση του Webserver. Για Webserver χρησιμοποιείται ο Apache, μαζί με το WSGI module, απαιτούμενο για την εκτέλεση των αρχείων Python του Project. Η διαδικασία που ακολουθείται από το πρώτο σενάριο είναι η εξής:

1. Εγκατάσταση του Apache, του WSGI module, της Python 3, του Python 3 PIP (υπεύθυνο για την διαχείρηση των modules που χρησιμοποιούνται από την Python), του Virtual Environment (`venv`) module της Python (υπεύθυνου για την απομόνωση του περιβάλλοντας εκτέλεσης του κώδικα από το υπόλοιπο σύστημα, προκειμένου να μην τροποποιηθούν ήδη υπάρχοντα πακέτα Python 3 του συστήματος<sup>[32]</sup>), καθώς επίσης και του cron (προκειμένου να καταστεί δυνατή η λειτουργία του συστήματος ειδοποιήσεων). Έπειτα από την εγκατάσταση των παραπάνω πακέτων, γίνεται ενεργοποίηση του WSGI module.
2. Αντιγραφή των αρχείων του Project στον κατάλογο εκτέλεσης (`/var/www/PiLock`) και αλλαγή ιδιοκτήτη και δικαιωμάτων στον κατάλογο και στα αρχεία αυτά. Ως ιδιοκτήτης και ομάδα ορίζονται ο `www-data`, καθώς αποτελεί τον χρήστη unix που χρησιμοποιείται από τον Apache για την εκτέλεση κώδικα. Η πρόσβαση ανακαλείται από τους υπόλοιπους χρήστες και ομάδες του συστήματος (`chmod 700`).
3. Γίνεται δημιουργία `venv` περιβάλλοντος και εγκατάσταση των απαιτούμενων module για την λειτουργία του PiLock σε αυτό (μέσω του αρχείου `requirements.txt` και του PIP).
4. Γίνεται δημιουργία και υπογραφή του πιστοποιητικού SSL που θα χρησιμοποιείται για την διασφάλιση της επικοινωνίας μεταξύ του Server και της εφαρμογής πελάτη.

Είναι σημαντικό να χρησιμοποιηθεί ως common name του πιστοποιητικού η τελική διεύθυνση στην οποία θα είναι ορατός ο Server.

5. Σε αυτό το σημείο πρέπει ο χρήστης να ενεργοποιήσει το πιστοποιητικό επεξεργάζοντας το αρχείο διαμόρφωσης του Apache υπεύθυνο για την χρήση του PiLock. Αυτό γίνεται επεξεργάζοντας το προεπιλεγμένο αρχείο διαμόρφωσης συνδέσεων SSL του Apache<sup>1</sup> και αλλάζοντας τις γραμμές `SSLCertificateFile` και `SSLCertificateKeyFile` με το νέο μιονοπάτι στο οποίο βρίσκονται το πιστοποιητικό και το κλειδί που δημιουργήθηκε προηγουμένως (`pilock.crt` και `pilock.key`) αντίστοιχα. Σε μία επόμενη έκδοση του PiLock θα αυτοματοποιηθεί πλήρως αυτή η εργασία, χωρίς να χρειάζεται ο χρήστης να δαπανήσει χρόνο.

Έπειτα από την εκτέλεση του πρώτου σεναρίου, και την ρύθμιση του πιστοποιητικού SSL, ο χρήστης πρέπει να εκτελέσει το δεύτερο αρχείο, γνωστό ως `setup_wsgi.sh`. Το αρχείο αυτό είναι υπεύθυνο για την ενεργοποίηση του PiLock configuration του Apache και την ρύθμιση του Django Project, συγκεκριμένα:

1. Γίνεται αντιγραφή του PiLock configuration στον κατάλογο διαμόρφώσεων του Apache.
2. Γίνεται το Database Migration. Μέσω αυτού του βήματος δημιουργείται η βάση δεδομένων και γίνεται αρχικοποίησή της.
3. Γίνεται συλλογή και αντιγραφή των στατικών αρχείων. Αυτά τα αρχεία είναι υπεύθυνα για την σωστή εμφάνιση του πίνακα διαχείρησης. Αποτελούνται κυρίως από αρχεία CSS, JavaScript και εικόνες.
4. Δημιουργία του πρώτου διαχειριστή του PiLock. Ζητούνται στοιχεία και από τα στοιχεία αυτά προκύπτει ο πρώτος διαχειριστής της πλατφόρμας.
5. Γίνεται αλλαγή χρήστη και δικαιωμάτων στο αρχείο της βάσης δεδομένων, προκειμένου να μην είναι δυνατή η ανάγνωσή της από άλλους χρήστες του συστήματος.
6. Γίνεται ενεργοποίηση του αρχείου διαμόρφωσης του PiLock που αντιγράφηκε στο βήμα 1 και επανεκκίνηση του Apache.
7. Προστίθεται ο χρήστης `www-data` στις ομάδες `dialout` και `gpio` προκειμένου να είναι εφικτή η πρόσβαση στην σειριακή θύρα και στο σύστημα GPIO αντίστοιχα.
8. Γίνεται αντιγραφή του cron file στο σύστημα.
9. Γίνεται επανεκκίνηση του συστήματος.

Έπειτα από την ολοκλήρωση της εκτέλεσης και του δεύτερου αρχείου, το σύστημα είναι έτοιμο για χρήση. Ο χρήστης μπορεί να βεβαιωθεί οτι λειτουργεί κάνοντας πλοιήγηση στην διεύθυνση στην οποία θα είναι ορατός ο Server. Αν ο Server στείλει μια απάντηση σε JSON, το σύστημα λειτουργεί σωστά.

Είναι σημαντικό να επισημανθεί οτι για να είναι ορατός ο Server εκτός του τοπικού δικτύου στο οποίο είναι εγκατεστημένος θα πρέπει να γίνει προώθηση θύρας (Port Forwarding) στον οικιακό δρομολογητή (Router) του χρήστη<sup>[33]</sup>.

---

<sup>1</sup>/etc/apache2/sites-available/000defaultssl.conf

# Κεφάλαιο 13

## Συνεχής Ενσωμάτωση - Continuous Integration

Η Συνεχής Ενσωμάτωση (Continuous Integration, CI), είναι μια πρακτική ανάπτυξης λογισμικού η οποία αναγκάζει τους Developers ενός Project να ανεβάζουν και να ενσωματώνουν κώδικα στο κύριο Branch ενός αποθετήριου, τουλάχιστον μία φορά την ημέρα. Παράλληλα με την ενσωμάτωση του κώδικα, γίνεται αυτοματοποιημένος έλεγχος (Automated Testing), μέσω Unit Testing ή/και Integration Testing. Αυτό καθιστά ευκολότερη την εύρεση και γρηγορότερη την διόρθωση των λαθών στον κώδικα, καθώς τα λάθη εντοπίζονται νωρίτερα καθ'όλη την διαδικασία της ανάπτυξης, μέσα σε λεπτά από την ενσωμάτωση του κώδικα στο αποθετήριο<sup>[39]</sup>.

Στο PiLock χρησιμοποιήθηκε Συνεχής Ενσωμάτωση μέσω Unit Tests, στον Server. Συγκεκριμένα, με κάθε νέο commit και Push στο αποθετήριο, στο GitHub, γίνεται η εξής σειρά ελέγχων, μέσω του Testing Framework του Django:

- Δημιουργείται μια βάση δεδομένων, αυστηρά για ελέγχους, καθώς επίσης και ένας χρήστης.
- Γίνεται Σύνδεση του χρήστη με το σύστημα του PiLock, χρησιμοποιώντας το πρώτο στάδιο.
- Γίνεται είσοδος με λανθασμένα στοιχεία (1ο στάδιο). Θα πρέπει να λάβει αρνητική απάντηση εισόδου.
- Γίνεται δοκιμή ξεκλειδώματος. Το σύστημα ελέγχει αν πήρε την απάντηση που έπρεπε να λάβει στην περίπτωση επιτυχούς ξεκλειδώματος. Αν είναι σωστή η απάντηση, το Test περνάει.
- Γίνεται δοκιμή ξεκλειδώματος με λανθασμένα στοιχεία. Συγκεκριμένα, γίνεται αλλαγή του τεκμηρίου πρόσβασης σε ένα τυχαίο τεκμήριο, τυχαίου μήκους. Θα πρέπει να λάβει απάντηση άρνησης πρόσβασης. Παρόμοια θα γίνει με το PIN και το νούμερο προφίλ συσκευής. Θα πρέπει σε όλες τις περιπτώσεις να λάβει αρνητικές απαντήσεις.

Τα παραπάνω Tests εκτελούνται μέσω μιας υπηρεσίας γνωστής ως Travis-CI<sup>1</sup>, η οποία αναλαμβάνει την κατασκευή νοητών μηχανών (Virtual Machines), προκειμένου να εκτελεστούν οι έλεγχοι.

---

<sup>1</sup><https://travis-ci.org/>

To παραπάνω Build Test γίνεται αυτόματα με κάθε νέο Commit που περνάει στο αποθετήριο μέσω Push. To Build Test θα πρέπει να περνάει υποχρεωτικά κατά την δι-αδικασία υποβολής Pull Request. Αν δεν περνάει το Build, δεν γίνεται να ενσωματωθεί το νέο branch πάνω στο Master.

# Κεφάλαιο 14

## Μελλοντική Επέκταση του PiLock

Το PiLock, όπως αναφέρθηκε προηγουμένως, αποτελεί ένα έργο ανοικτού κώδικα, στο οποίο μπορούν όσοι χρήστες θέλουν, να προτείνουν ή να υλοποιήσουν νέα λειτουργικότητα ή να διορθώσουν προβλήματα στον ήδη υπάρχον κώδικα. Αξίζει όμως να αναφερθούν τρόποι επέκτασης του με νέα λειτουργικότητα ή οποία μπορεί να προστεθεί στο μέλλον, σε επόμενες εκδόσεις του λογισμικού. Πέρα των παρακάτω πιθανών επεκτάσεων που θα αναλυθούν στο κεφάλαιο, είναι σημαντικό να τονιστεί οτι μπορούν να προτείνουν οι χρήστες παραπάνω επεκτάσεις λειτουργικότητας, ανοίγοντας ένα Issue στο επίσημο αποθετήριο του PiLock στο GitHub.

### 14.1 Σύνδεση με Κεντρικό Σημείο Ελέγχου 'Εξ-υπνων Συσκευών

Πολλές εγκαταστάσεις έξυπνων συσκευών σπιτιών παρέχουν ένα κεντρικό σημείο ελέγχου (Hub) όλων των συσκευών που είναι εγκατεστημένες μέσα στο σπίτι. Μπορεί, μέσω ενός μελλοντικού Update, να επεκταθεί το API του PiLock προκειμένου να προσφέρει λειτουργικότητα συνεργασίας με κάποιο επώνυμο ή μη Hub ελέγχου συσκευών. Για να γίνει αυτό, θα πρέπει να γίνει προσεκτική επιλογή κάποιας πλατφόρμιας, αν πρόκειται να γίνει συμβατό με μόνο μία πλατφόρμα ελέγχου συσκευών, και κατόπιν να ακολουθηθεί από προσεκτικό σχεδιασμό και υλοποίηση, ή να γίνει ακόμα πιο προσεκτικός σχεδιασμός, σε περίπτωση που επιλεχθούν παραπάνω από μία πλατφόρμες ελέγχου, προκειμένου να μην υπάρξουν προβλήματα στο τελικό έργο, και για να είναι εξίσου καλό το αποτέλεσμα για όλες τις πλατφόρμες που θα στοχεύει.

Πιθανές πλατφόρμες για να γίνει επέκταση αποτελούν το Google Home, το Apple Home, το Amazon Alexa και το OpenHAB, που αποτελεί Δωρεάν και Ανοικτού Κώδικα Λογισμικό.

### 14.2 Ξεκλείδωμα μέσω αναγνώρισης δακτυλικού αποτυπώματος

Εφόσον υπάρχει ήδη μηχανισμός για ξεκλειδώματα χωρίς PIN, μπορεί να τον εκμεταλλευτεί ένα σύστημα ξεκλειδώματος που να χρησιμοποιεί το δακτυλικό αποτύπωμα του χρήστη

προκειμένου να επιβεβαιώσει την ταυτότητα του χρήστη και να αποχρυπωγραφήσει το Auth Token, που χρησιμοποιείται για το ξεκλείδωμα. Το σύστημα αυτό χρησιμοποιεί τις ενσωματωμένες στα Android (από την έκδοση 6 και μετά) λειτουργίες αυθεντικοποίησης, και εγγυάται πολύ μεγαλύτερη ασφάλεια από το συμβατικό σύστημα με PIN που είναι ήδη υλοποιημένο, καθώς τα αποτυπώματα των χρηστών είναι μιοναδικά και είναι δύσκολη η αντιγραφή τους<sup>[34]</sup>.

### 14.3 Ενσωμάτωση αυθεντικοποίησης μέσω LDAP, AD DS

Εφόσον κάποιος οργανισμός επιλέξει να χρησιμοποιήσει το PiLock ως σύστημα ελέγχου πρόσβασης σε διάφορα σημεία του, είναι πολλές φορές επιθυμητό να χρησιμοποιούνται ήδη υπάρχοντα συστήματα αυθεντικοποίησης όπως το Lightweight Directory Access Protocol (LDAP) ή το Active Directory Directory Services (AD DS), που χρησιμοποιούνται από τον οργανισμό προκειμένου οι χρήστες του να αυθεντικοποιούνται σε διάφορα συστήματά του. Για να γίνει αυτό, θα πρέπει να φτιαχτεί ένα module που σκοπός του θα είναι να συνεργάζεται με αυτά τα συστήματα και να αντλεί τις ομάδες που θα του ορίζει ο διαχειριστής προκειμένου να έχουν πρόσβαση μόνο εξουσιοδοτημένες ομάδες χρηστών στο PiLock. Με αυτό τον τρόπο αυθεντικοποίησης των χρηστών αποφεύγεται η ανάγκη εκ νέου εγγραφής χρηστών ενός οργανισμού στο σύστημα του PiLock, έπειτα από την εγκατάσταση, και συγχρονίζεται αυτόματα όταν προστίθενται νέοι χρήστες.

### 14.4 Ξεκλείδωμα Επισκεπτών

Προτάθηκε από τον Χαράλαμπο Νικολάτο τον Αύγουστο του 2017.

Πολλές φορές είναι επιθυμητό να μπορεί ο χρήστης να καλέσει κάποιο φίλο ή συγγενή στο σπίτι του. Για να διευκολυνθεί η πρόσβαση του/των μπορεί να υλοποιηθεί ένα Σύστημα Ξεκλειδώματος Προσκεκλημένων/Επισκεπτών (Guest Unlock). Αφότου ο διαχειριστής εισάγει την διεύθυνση ηλεκτρονικού ταχυδρομείου του επισκέπτη στο PiLock, θα δημιουργείται ένας μιοναδικός κωδικός αρκετά μεγάλου μήκους (16 χαρακτήρες, πεζά, κεφαλαία γράμματα, αριθμοί) και θα αποστέλλεται στην διεύθυνση του επισκέπτη που εισήχθη από τον διαχειριστή. Έπειτα, θα μπορεί ο επισκέπτης, αντιγράφοντας τον κωδικό αυτόν, να τον χρησιμοποιήσει για ένα μιοναδικό ξεκλείδωμα μέσω της εφαρμογής του PiLock. Έπειτα από την χρήση του, ο κωδικός θα διαγράφεται από το σύστημα και θα είναι αδύνατον να επαναχρησιμοποιηθεί.

# Κεφάλαιο 15

## Επίλογος - Συμπεράσματα

Μέσα από όλα τα προηγούμενα κεφάλαια αναδείχτηκε η διαδικασία του σχεδιασμού και της υλοποίησης του PiLock, και παράλληλα περιγράφηκε εξονυχιστικά ο τρόπος λειτουργίας του. Έπειτα από όλη την διαδικασία αυτή, εξάχθηκαν συγκεκριμένα συμπεράσματα ως προς τον σχεδιασμό τέτοιου είδους έργων, είτε έχουν να κάνουν μιες έξυπνες σπίτια/συσκευές, είτε μιες ενσωματώση διάφορων συσκευών στο διαδίκτυο προκειμένου να είναι δυνατόν να ελεγχθούν.

Πρώτο και κυριότερο συμπέρασμα είναι ότι τέτοιου είδους συστήματα πρέπει να λειτουργούν με κατάλληλα σχεδιασμένα πρότυπα και δικλείδες ασφαλείας, προκειμένου να αποφευχθούν επιθέσεις τρίτων, ειδικά αν πρόκειται για σύστημα εξουσιοδότησης πρόσβασης, όπως το PiLock.

Το PiLock, καθώς πρόκειται για λογισμικό ανοικτού κώδικα, κληρονομεί τα προτερήματα των λογισμικών αυτού του είδους, και του παρέχεται δυνατότητα ανάπτυξης λειτουργικότητας και διόρθωσης λαθών από τους ίδιους του τους χρήστες, καθώς επίσης και, όπως αναφέρθηκε στην ενότητα 5.1, μπορεί εξαιτίας της ανοικτότητάς του να χρησιμοποιείται για οποιοδήποτε σκοπό, είτε από ιδιότες, είτε από εταιρίες, και αυτό χωρίς κανένα κόστος.

Αναλύθηκε επίσης η καινοτομία του PiLock στις φορετές συσκευές, καθώς, όπως αναφέρθηκε στην ενότητα 1.4.3, είναι ένα από τα πρώτα οικιακά συστήματα πρόσβασης παγκοσμίως που χρησιμοποιεί Android Wear Smartwatch προκειμένου να ξεκλειδώνεται η πόρτα, και έπειτα από ενδελεχή έρευνα, πιθανόν να είναι και το πιο εξελιγμένο, μη εμπορικό σύστημα πρόσβασης στην κατηγορία αυτή. Το χαμηλό του κόστος, τα προσιτά του υλικά και η εύκολή του εγκατάσταση το κάνουν εύχρηστο και ιδανικό για το μεγαλύτερο ποσοστό των χρηστών του.

Είναι σημαντικό καθ'όλη την διαδικασία της ανάπτυξης ενός λογισμικού να χρησιμοποιείται κάποιο λογισμικό διαχείρησης εκδόσεων, προκειμένου να διαχειρίζεται αποτελεσματικά ο νέος κώδικας ο οποίος θα εισαχθεί στο λογισμικό, αλλά και για να χρατείται απομονωμένος από τον ήδη ενεργό κώδικα. Στην ενότητα 5.3.1, αναλύθηκαν τα συστήματα διαχείρησης εκδόσεων καθώς επίσης και οι κανόνες που επινοήθηκαν προκειμένου να είναι ασφαλής η ανάπτυξη του PiLock, και να γίνεται αποτελεσματική διαχείριση του κώδικα του. Στο Κεφάλαιο 13 αναλύθηκε η αναγκαιότητα χρήσης συστημάτων Συνεχούς Ενσωματώσης, μέσω των οποίων επιτυγχάνεται αποτελεσματικός έλεγχος του κώδικα, καθώς επίσης και όσο το δυνατόν γρηγορότερη εύρεση λαθών και διόρθωσή τους.

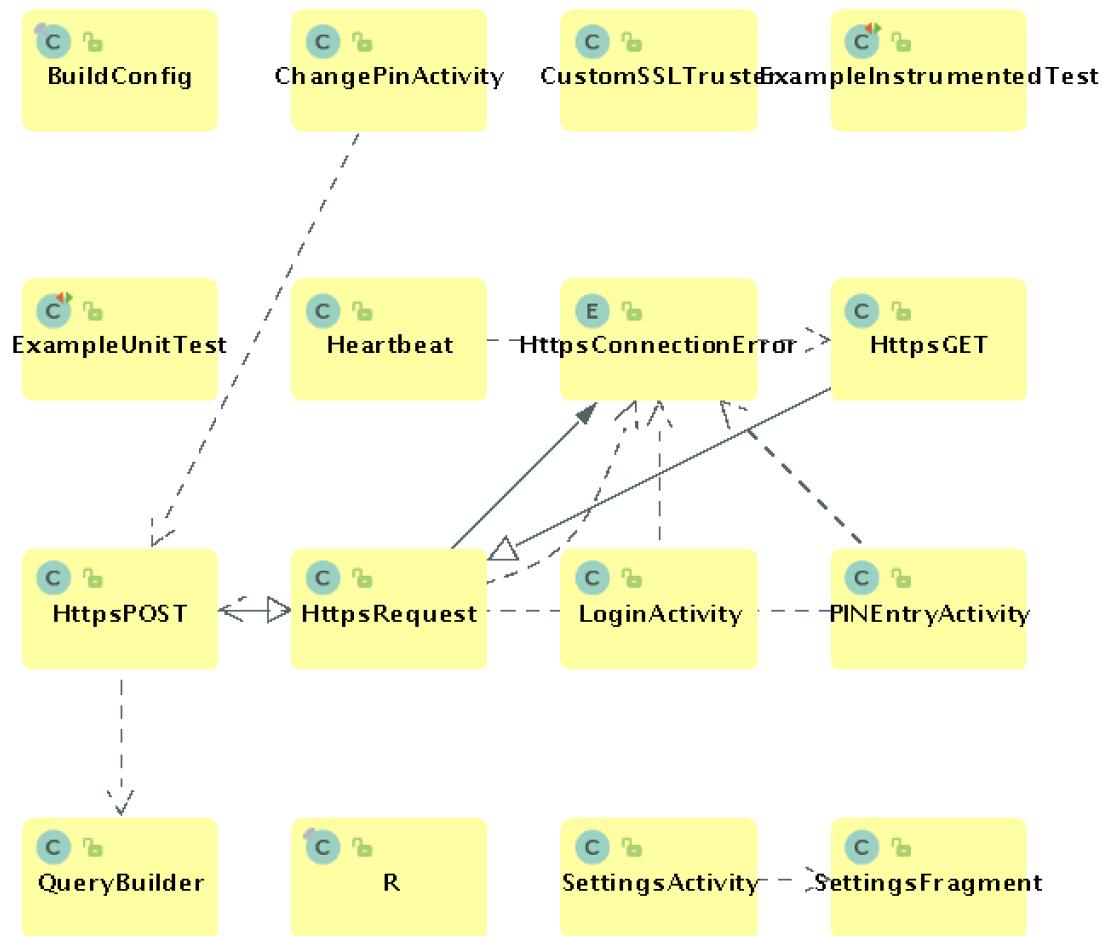
Τέλος, στο Κεφάλαιο 14, αναλύθηκαν διάφοροι τρόποι επέκτασης του PiLock προκειμένου να προστεθούν χρήσιμες λειτουργίες σε αυτό (όπως η λειτουργία ξεκλειδώματος επισκεπτών), να βελτιωθεί η ασφάλεια του (ξεκλείδωμα με χρήση δακτυ-

λικού αποτυπώματος), να ενσωματωθεί με ήδη υπάρχοντα συστήματα έξυπνων σπιτιών, καθώς επίσης και με εταιρίες (ενσωμάτωση LDAP, AD DS). Μέσω αυτών των προσθηκών, θα συνεχίσει να εξελίσσεται και θα γίνει ακόμα πιο εύκολο στην χρήση και ασφαλές.

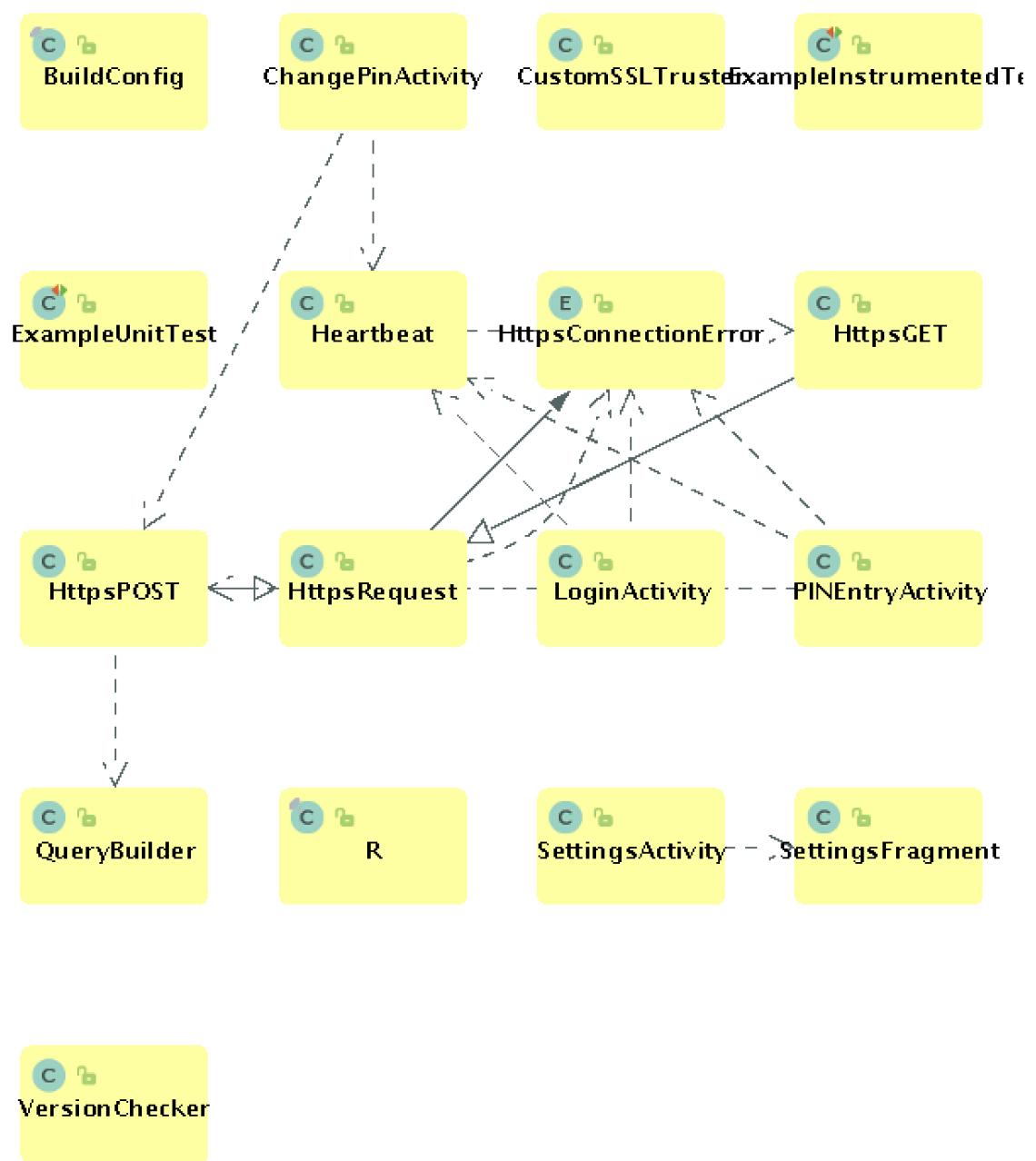
# Appendix A

## Διαγράμματα

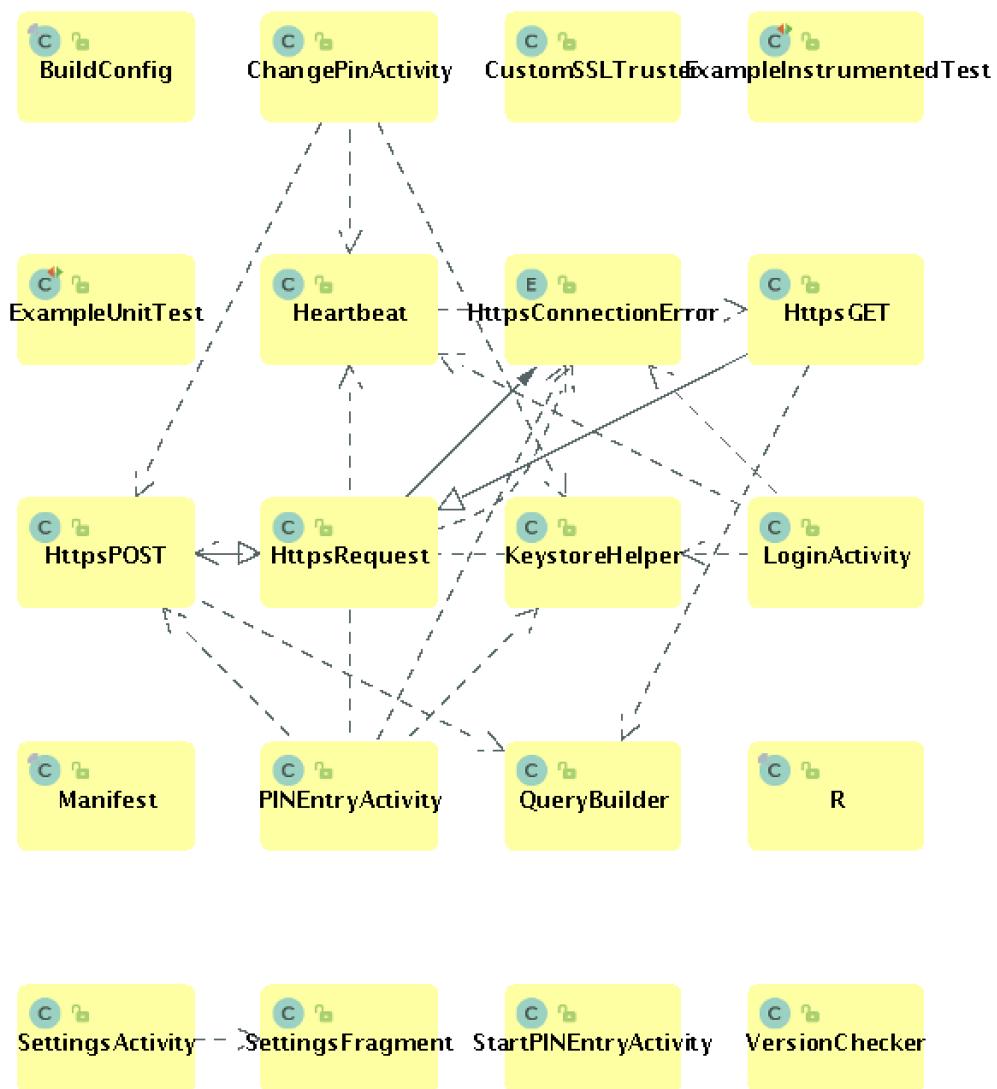
### A.1 Συνοπτικά Διαγράμματα Κλάσεων Εκδόσεων



Εικόνα A.1: Συνοπτικό διάγραμμα κλάσεων για την έκδοση 0.1.0



Εικόνα A.2: Συνοπτικό διάγραμμα κλάσεων για την έκδοση 0.2.0

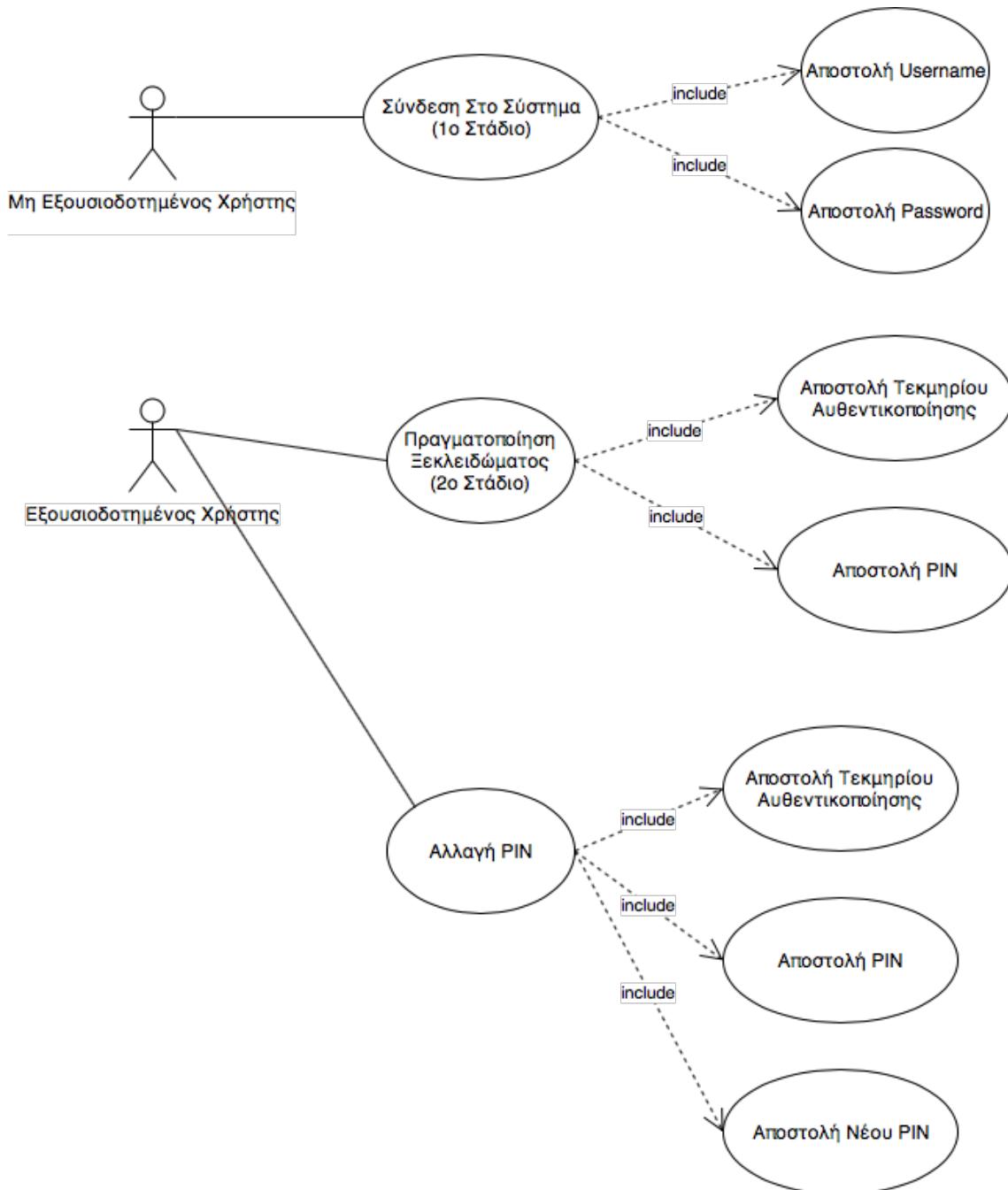


Εικόνα A.3: Συνοπτικό διάγραμμα κλάσεων για την έκδοση 0.3.0

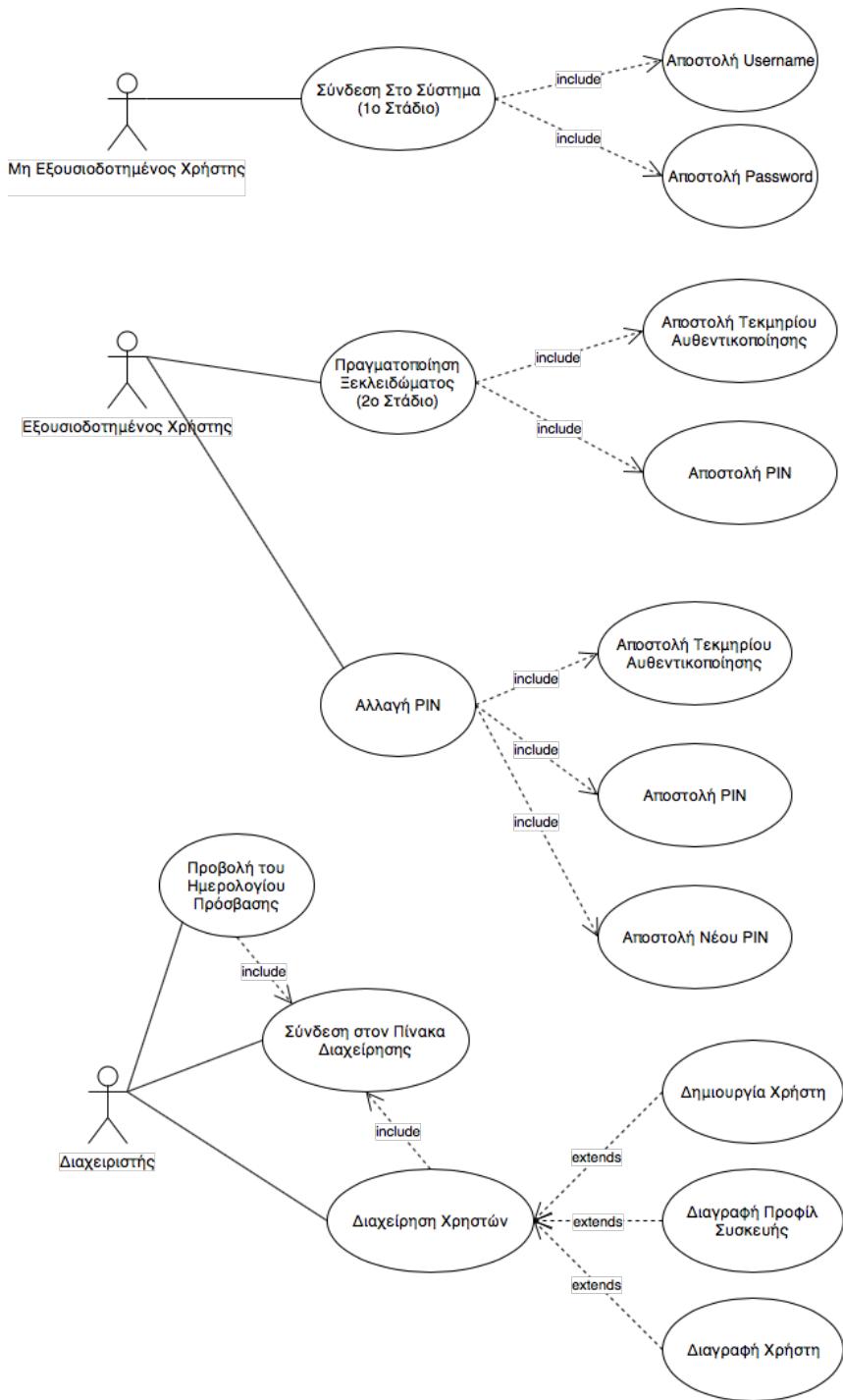


Εικόνα A.4: Συνοπτικό διάγραμμα κλάσεων για την έκδοση για Android Wear

## A.2 Διαγράμματα Περιπτώσεων Χρήσεων Εκδόσεων



Εικόνα A.5: Διάγραμμα Περιπτώσεων Χρήσης για την έκδοση 0.1.0



Εικόνα A.6: Διάγραμμα Περιπτώσεων Χρήστης για την έκδοση 0.2.0



Εικόνα A.7: Διάγραμμα Περιπτώσεων Χρήσης για την έκδοση 0.3.0

# Βιβλιογραφεία

- [1] Kevin Ashton (2009), "That 'Internet of Things' thing"  
<http://www.rfidjournal.com/articles/view?4986>
- [2] Jim Hill (2015), "The smart home: a glossary guide for the perplexed"  
<https://www.t3.com/features/the-smart-home-guide>
- [3] Ian Paul (2017), "The \$10 Raspberry Pi Zero W brings Wi-Fi and Bluetooth to the minuscule micro-PC"  
<https://www.pcworld.com/article/3175256/computers/the-10-raspberry-pi-zero-w-brings-wi-fi-and-bluetooth-to-the-minuscule-micro-pc.html>
- [4] Eben Upton (2015), "RASPBERRY PI ZERO: THE \$5 COMPUTER"  
<https://www.raspberrypi.org/blog/raspberry-pi-zero/>
- [5] Relay, Wikipedia  
<https://en.wikipedia.org/wiki/Relay>
- [6] Arduino for Beginners, Makerspaces.com  
<https://www.makerspaces.com/arduino-uno-tutorial-beginners/>
- [7] Jump Wire Structure (2003), Katayama Tatsuo  
<http://www.freepatentsonline.com/6899560.html>
- [8] How to connect raspberry pi to WiFi without a monitor (2017), Chetan Kapoor  
<https://installvirtual.com/how-to-connect-raspberry-pi-to-wifi-without-a-monitor>
- [9] A security update for Raspbian PIXEL (2016), Simon Long  
<https://www.raspberrypi.org/blog/a-security-update-for-raspbian-pixel/>
- [10] <https://www.raspberrypi.org/documentation/linux/usage/users.md>
- [11] How to Create a Secure Password You Can Remember Later: 4 Key Methods (2014), Kevan Lee  
<https://open.buffer.com/creating-a-secure-password/>
- [12] Updating and Upgrading Raspbian <https://www.raspberrypi.org/documentation/raspbian/updating.md>
- [13] <https://www.raspberrypi.org/documentation/remote-access/ip-address.md>
- [14] Τι είναι το λογισμικό ανοικτού κώδικα: Μια εισαγωγή (2015), Κώστας Παπαδήμας  
<https://ellak.gr/2015/09/ti-ine-to-logismiko-aniktou-kodika-mia-isagogi/>

- [15] 10 Reasons Open Source Is Good for Business (2010), Katherine Noyes  
[https://www.pcworld.com/article/209891/10\\_reasons\\_open\\_source\\_is\\_good\\_for\\_business.html](https://www.pcworld.com/article/209891/10_reasons_open_source_is_good_for_business.html)
- [16] Free and Open Source, About, git-scm.com  
<https://git-scm.com/about/free-and-open-source>
- [17] Getting Started - A Short History of Git, git-scm.com  
<https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>
- [18] Data Assurance, About, git-scm.com  
<https://git-scm.com/about/info-assurance>
- [19] About Bootstrap, getbootstrap.com  
<https://getbootstrap.com/docs/3.3/about/>
- [20] Bootstrap in A List Apart No. 342 (2012), Mark Otto  
<http://markdotto.com/2012/01/17/bootstrap-in-a-list-apart-342/>
- [21] What is jQuery, jquery.com  
<https://jquery.com/>
- [22] git-cola Licence  
<https://git-cola.github.io/license.html>
- [23] Why passwords should be hashed (2011), Thomas Pornin  
<https://security.blogoverflow.com/2011/11/why-passwords-should-be-hashed/>
- [24] Included Validators - Password Management in Django, Django 2.0 Documentation  
<https://docs.djangoproject.com/en/2.0/topics/auth/passwords/#includedValidators>
- [25] HTTP Strict Transport Security (RFC-6797) (2012), Internet Engineering Task Force (IETF)  
<https://tools.ietf.org/html/rfc6797#section-2.3>
- [26] Raspberry Pi GPIO Pinout, Phil Howard  
<https://pinout.xyz/>
- [27] Serial Port, Arduino Reference, arduino.cc  
<https://www.arduino.cc/reference/en/language/functions/communication/serial/>
- [28] Activity, Android Reference, Android Developers  
<https://developer.android.com/reference/android/app/Activity>
- [29] Android Secure Shared Preferences, Gaurav Vashisth (2015)  
<https://medium.com/@vashisthg/android-secure-shared-preferences-10f8356a4c2b>
- [30] Android Keystore, Android Reference, Android Developers  
<https://developer.android.com/training/articles/keystore>

- [31] Send and Recieve Messages on Wear, Android Developers  
<https://developer.android.com/training/wearables/data-layer/messages>
- [32] Introduction, Virtualenv Documentation  
<https://virtualenv.pypa.io/en/stable/#introduction>
- [33] Definition: Port Forwarding, PC Magazine  
<https://www.pc当地.com/encyclopedia/term/49509/port-forwarding>
- [34] How to add fingerprint authentication to your android app (2017), Jessica Thornsby  
<https://www.androidauthority.com/how-to-add-fingerprint-authentication-to-your->
- [35] The Best Smart Locks of 2018 (2018), John R. Delaney  
<https://www.pc当地.com/article/344336/the-best-smart-locks>
- [36] About the Django Software Foundation  
<https://www.djangoproject.com/foundation/>
- [37] Django Philosophies, Django Documentation  
<https://docs.djangoproject.com/en/2.0/misc/design-philosophies/>
- [38] random.SystemRandom, Python Standard Library Documentation  
<https://docs.python.org/2/library/random.html#random.SystemRandom>
- [39] Continuous Integration, Explained, Dan Radigan, atlassian.com  
<https://www.atlassian.com/continuous-delivery/continuous-integration-intro>
- [40] GNU General Public Licence, version 3 (2007)  
<https://www.gnu.org/licenses/gpl-3.0.en.html>

# Index

- Σειριακή Θύρα (Serial Port), 9
- Εφαρμογές (Django Apps), 35
- Εξουσιοδοτημένη Συσκευή, 37
- Εξουσιοδοτημένος χρήστης, 37
- Μη εξουσιοδοτημένος χρήστης, 37
- Μη Εξουσιοδοτημένη Συσκευή, 37
- Active Directory Directory Services (AD DS), 58
- AdminCP, 35
- AdminCP (Administration Control Panel), 50
- Android Studio, 25
- Arduino Nano, 9
- Arduino UNO, 9
- Auth Token (Authorization Token), 35
- Branching System, 22
- Business Logic, 24
- Commits, 22
- CSS3 (Cascading Style Sheets), 24
- Debug Mode, 51
- Django Project, 35
- Domotics, 5
- Git, 22
- git-cola, 26
- GitHub, 23
- GitLab, 23
- GNU Nano, 26
- GNU Privacy Guard (GPG), 22
- HTML5 (HyperText Markup Language), 24
- Internet of Things, 4
- Issue Tracking, 23
- JavaScript, 24
- Lightweight Directory Access Protocol (LDAP), 58
- Merge Request, 23
- Milestone, 24
- Model-Template-View Architecture - MTV, 34
- One-To-One Relationship, 35
- OTG Cable, 10
- Personal Identification Number (PIN), 35
- PiLock Administration Control Panel (PiLock AdminCP), 5
- Profile (Device Profile), 35
- Pull Request, 23
- Raspberry Pi Zero W (RPi Zero W), 8
- Relay Module, 9
- Shell, 24
- Staging, 22
- Sublime Text, 26
- Wear Token, 39
- Wiring, 24
- XML (Cross Markup Language), 24
- YAML, 24