



ΔΗΜΙΟΥΡΓΙΑ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΤΗΝ ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΗΣ ΠΑΝΔΗΜΙΑΣ COVID-19

Φοίβος Αργυρίδης
Π2013041

Επιβλέπων καθηγητής: Αυλωνίτης Μάρκος

Περιεχόμενα

Πίνακες	2
Εικόνες	2
Περίληψη.....	3
Abstract	3
Εισαγωγή	4
1. Εφαρμογές για COVID-19	5
1.1. Εφαρμογές ιχνηλάτησης στην Ευρωπαϊκή Ένωση	6
1.2. Contact tracing. Apple & Google	8
1.3. Εφαρμογές ιχνηλάτησης και προσωπικά δεδομένα.....	12
2. Δημιουργία ενημερωτικής εφαρμογής για τον covid-19.....	13
2.1. Επιλογή τεχνολογιών.....	14
2.2. Πηγές δεδομένων.....	15
2.3. Λογότυπο και όνομα εφαρμογής.....	15
2.4. Περιγραφή λειτουργίας web app.....	16
2.5. Database schema	21
2.6. Code structure	23
2.6.1. Models	23
2.6.2. Controllers	25
2.6.3. Migrations	32
2.6.4. Routes.....	34
2.6.5. Middlewares	35
2.6.6. Views	35
2.7. Περιγραφή λειτουργίας mobile application.....	36
2.8. Άδειες εφαρμογής.....	41
2.9. 3 rd party libraries	41
2.10. Εκδόσεις android που υποστηρίζει η εφαρμογή.....	41
2.11. File and code structure εφαρμογής	42
2.12. Other resources.....	49
3. Προτάσεις βελτίωσης.....	49
4. Συμπεράσματα	50
Βιβλιογραφία	52

Πίνακες

Mobile contact tracing apps in EU Member States (European Union, 2021)	7
Κράτη που έχουν δημιουργήσει εφαρμογές βασισμένες στο σύστημα ειδοποιήσεων της Apple - Google (xda developers, 2021)	12

Εικόνες

Αναπαράσταση λειτουργίας του συστήματος ειδοποιήσεων Apple – Google (Google, 2020) 9	
Mobile Operating System Market Share Worldwide (statcounter, 2021)	14
Λογότυπο πρόταση 1	15
Λογότυπο πρόταση 2	15
Αρχική εικόνα εφαρμογής.....	16
Διαγραμματική απεικόνιση αποτελεσμάτων έρευνας	16
Login εφαρμογής.....	17
Διαγραφή χρήστη.....	17
Δημιουργία νέου ερωτηματολογίου.....	18
Δημιουργία νέου χρήστη	19
Προφίλ χρήστη	19
Διαχείριση ερωτηματολογίων (Admins - Super admin).....	20
Διαγραφή ερωτηματολογίου.	20
Παραδειγμα δομής του data array	28
Οθόνη εφαρμογής με τα στοιχεία της πανδημίας.....	37
Προσωπικές πληροφορίες χρήστη.....	38
Ερωτηματολόγιο.	38
Επιτυχής υποβολή απαντήσεων.	39
Ο χρήστης έχει ήδη απαντήσει στο ερωτηματολόγιο.	39
Πληροφορίες εφαρμογής.....	40

Περίληψη

Σκοπός της εργασίας είναι η ανάπτυξη εφαρμογής για την αντιμετώπιση της πανδημίας covid-19. Αρχικά αναλύονται παρόμοιες εφαρμογές και προβλήματα υλοποίησης τους, με έμφαση την προστασία των προσωπικών δεδομένων των χρηστών σε διάφορες χώρες του πλανήτη. Σε δεύτερο επίπεδο παρουσιάζεται η επιλογή των τεχνολογιών για τη δημιουργία της εφαρμογής καθώς και τα στάδια υλοποίησης της. Καθώς πρόκειται για εφαρμογή ανοιχτού κώδικα αναλύεται και τρόπος δημιουργίας (code analysis, folder structure κ.λ.π.). Στο τέλος κάνουμε προτάσεις βελτίωσης και επέκτασης της εφαρμογής και αναλύουμε τα συμπεράσματα που προκύπτουν.

Abstract

The goal of this paper is the development of a mobile app to help fight covid-19. The paper analyses similar apps and their problems in the scope of users' privacy. The paper continues presenting the technologies that was used for the creation of the app. Since it is an open-source app, the paper analyses the creation methods (code analysis, folder structure, etc.). Finally, we suggest possible improvements and extension for the app. We analyze conclusions.

Keywords

Covid-19, mobile apps, contact tracing apps, Laravel, android studio, Firebase, privacy protection, data privacy protection, informational apps

Εισαγωγή

Η πανδημία covid-19 (coronavirus disease 2019) βρήκε τα συστήματα υγείας στα περισσότερα μέρη του πλανήτη απροετοίμαστα. Καθώς δεν υπήρχε αποτελεσματικό μέσο θεραπείας και πρόληψης, η σύσταση στο γενικό πληθυσμό ήταν και είναι η τήρηση μέτρων προστασίας όπως η χρήση μάσκας, η κοινωνική αποστασιοποίηση, τακτικό πλύσιμο χεριών και απολύμανση επιφανειών. Σε πάρα πολλές περιπτώσεις επιβλήθηκαν αυστηρά lockdown καθώς υπήρχαν αρκετά κρούσματα με αποτέλεσμα την πίεση των συστημάτων υγείας.

Η χρήση της τεχνολογίας μπορεί να συμβάλει στην ενημέρωση του πληθυσμού, στην καλύτερη ανίχνευση κρουσμάτων και στην κατανόηση των ψυχολογικών και κοινωνικών επιπτώσεων από την κοινωνική αποστασιοποίηση και τα lockdown. Στόχος της συγκεκριμένης εργασίας είναι η δημιουργία εφαρμογής για κινητό τηλέφωνο όπου χρήστης θα μπορεί να ενημερώνεται για την εξέλιξη της πανδημίας, καθώς και να απαντά σε έρευνες. Για την δημιουργία της εφαρμογής θα ανιχνευτούν και θα αναλυθούν παρόμοιες εφαρμογές.

1. Εφαρμογές για COVID-19

Από την αρχή της πανδημίας δημιουργήθηκαν αρκετές εφαρμογές για κινητές συσκευές για να βοηθήσουν στη μείωση της εξάπλωσης του ιού. Οι πρώτες εφαρμογές στόχευαν στην ειδοποίηση των χρηστών σε περίπτωση που είχαν έρθει σε επαφή με επιβεβαιωμένο κρούσμα και ονομάζονται **contact tracing apps** (εφαρμογές ιχνηλάτησης) . Η πρώτη επίσημη κρατική εφαρμογή δημιουργήθηκε από την κυβέρνηση της Σιγκαπούρης και εκμεταλλευόταν την τεχνολογία Bluetooth για να ανταλλάσσει πληροφορίες στο παρασκήνιο με άλλες συσκευές. Έτσι αν κάποιος δηλώσει ότι έχει μολυνθεί αυτόματα στέλνονται push notifications στις συσκευές που είχαν έρθει σε κοντινή απόσταση με τον χρήστη τις τελευταίες μέρες. Η συγκεκριμένη τεχνολογία έγινε ανοιχτού κώδικα και την χρησιμοποίησαν και άλλα κράτη για την ανάπτυξη παρόμοιων εφαρμογών. (Singh, Couch & Yap, 2020).

Παράλληλα με τις εφαρμογές ιχνηλάτησης δημιουργήθηκαν και εφαρμογές παρακολούθησης συμπτωμάτων. Οι εφαρμογές αυτές παρακολουθούν την υγεία του χρήστη είτε με ερωτηματολόγια όπου ο χρήστης συμπληρώνει μέσω της εφαρμογής σε τακτά χρονικά διαστήματα είτε από άλλες συσκευές wearables οι οποίες καταγράφουν τους καρδιακούς παλμούς, τη θερμοκρασία και το οξυγόνο στο αίμα. Ο χρήστης λαμβάνει ειδοποίηση για πιθανή μόλυνση. (Singh et al., 2020)

Αρκετές χώρες δημιούργησαν εφαρμογές καραντίνας με σκοπό τον περιορισμό των ατόμων που έχουν βρεθεί θετικοί στον ιό. Οι εφαρμογές αυτές καταγράφουν την τοποθεσία του χρήστη που είναι υποχρεωμένος να μείνει σε καραντίνα, Αν ο χρήστης παραβιάσει την καραντίνα ενημερώνονται αυτόματα οι αρχές. Η Ινδία ήταν από τις πρώτες χώρες που χρησιμοποίησαν τέτοιου είδους εφαρμογές, ενώ στο Hong Kong όσοι μπαίνουν στη χώρα είναι υποχρεωμένοι να φοράνε ειδική συσκευή στο χέρι που σε συνεργασία με ειδική εφαρμογή καταγράφουν συνεχώς την τοποθεσία του χρήστη και σε περίπτωση που αυτός σπάσει την καραντίνα ειδοποιούνται αυτόματα οι αρχές. Πολλές χώρες έχουν κάνει υποχρεωτική τη χρήση τέτοιων εφαρμογών σε άτομα που έχουν βρεθεί θετικά στον ιό. (Singh et al., 2020).

1.1. Εφαρμογές ιχνηλάτησης στην Ευρωπαϊκή Ένωση

Πολλά κράτη μέλη της Ευρωπαϊκής ένωσης έχουν αναπτύξει εφαρμογές ιχνηλάτησης για την ενημέρωση των ατόμων σε περίπτωση που έρθουν σε επαφή με κρούσμα. Οι εφαρμογές έχουν στόχο να διευκολύνουν το δύσκολο έργο της ιχνηλάτησης πιθανών κρουσμάτων. Σε περίπτωση που ο κάτοχος της εφαρμογής δηλώσει ότι έχει μολυνθεί με τον ιό, όσοι έχουν εγκατεστημένη την εφαρμογή και έχουν έρθει σε επαφή μαζί του λαμβάνουν ειδοποίηση για να εξεταστούν για πιθανή μόλυνση. (European Union, 2021). Τα περισσότερα κράτη μέλη διαθέτουν τέτοιες εφαρμογές και η πλειοψηφία αυτών μπορεί να επικοινωνεί ή θα μπορεί να επικοινωνεί στο μέλλον και με άλλες εφαρμογές (interoperable ή potentially interoperable). Τα κράτη μέλη που δεν διαθέτουν ακόμα εφαρμογή είναι η Βουλγαρία, η Ελλάδα, το Λουξεμβούργο, η Ρουμανία, η Σλοβακία και η Σουηδία. Στον παρακάτω πίνακα φαίνονται αναλυτικά οι εφαρμογές του κάθε κράτους. Οι εφαρμογές είναι διαθέσιμες σε Google Play (Android) και App Store (Ios). (European Union, 2021).

Countries	App	Interoperable - is this app potentially interoperable?	Interoperable - can this app already talk to another app?
Austria	Stopp Corona App	Yes	Yes
Belgium	Coronalert	Yes	Yes
Bulgaria	The deployment of a contact tracing app is not foreseen.		
Croatia	Stop COVID-19	Yes	Yes
Cyprus	CovTracer-EN	Yes	Yes
Czechia	eRouška	Yes	Yes
Denmark	Smittestop	Yes	Yes
Estonia	HOIA	Yes	No
Finland	Koronavilkku	Yes	Yes
France	TousAntiCovid	No	No

Germany	Corona-Warn-App	Yes	Yes
Greece	Contact tracing app under development.	Yes	
Hungary	VirusRadar	No	No
Ireland	COVID Tracker	Yes	Yes
Italy	Immuni	Yes	Yes
Latvia	Apturi Covid	Yes	Yes
Lithuania	Korona Stop LT	Yes	No
Luxembourg	The deployment of a contact tracing app is not foreseen.		
Malta	COVIDAlert	Yes	Yes
Netherlands	CoronaMelder	Yes	Yes
Norway	Smittestopp	Yes	Yes
Poland	ProteGO Safe	Yes	Yes
Portugal	StayAway COVID	Yes	No
Romania	Romania is exploring the development of a contact tracing app.	Yes	No
Slovakia	A contact tracing app is being developed.		
Slovenia	#OstaniZdrav	Yes	Yes
Spain	Radar Covid	Yes	Yes
Sweden	The deployment of a contact tracing app is not foreseen.		

Πίνακας 1 Mobile contact tracing apps in EU Member States (European Union, 2021)

1.2. contact tracing. Apple & Google

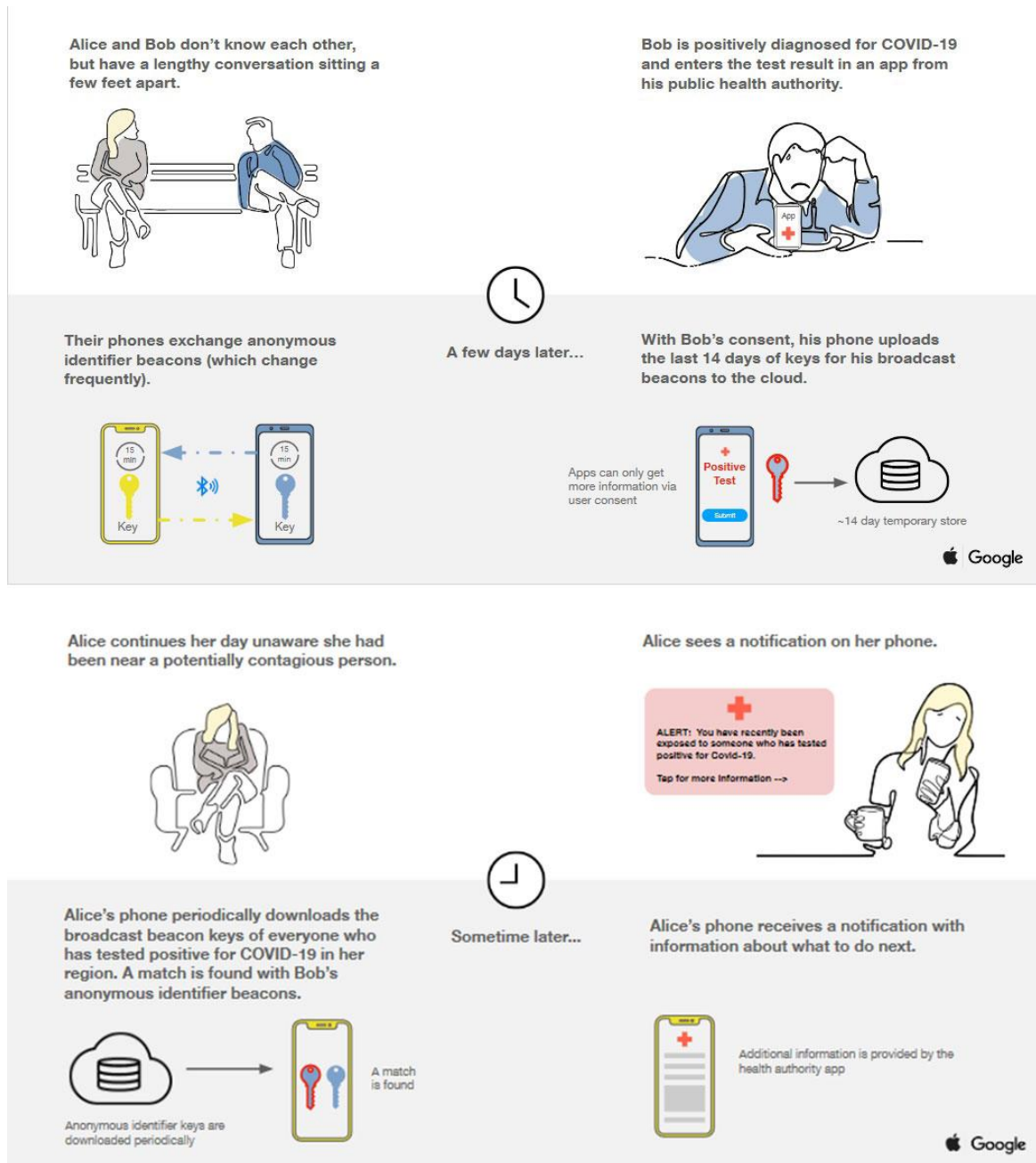
Σε παγκόσμιο επίπεδο η Google με την Apple ένωσαν τις δυνάμεις τον Μάιο του 2020 δημιουργώντας APIS τα οποία επιτρέπουν τον διαμοιρασμό πληροφοριών μεταξύ συσκευών ANDROID και IOS όταν αυτά χρησιμοποιούν επίσημες κρατικές εφαρμογές για την αποφυγή της μετάδοσης του ιού. Σε δεύτερη φάση ενσωμάτωσαν στα λειτουργικά τους συστήματα το σύστημα ειδοποιήσεων και οι χρήστες έχουν την επιλογή να το ενεργοποιήσουν. (Apple 2020)

Η επιτυχία των εφαρμογών αυτών εξαρτάται από την ποσότητα των χρηστών. Όσοι περισσότεροι κατεβάσουν την εφαρμογή τόσο πιο αποτελεσματική θα είναι η ιχνηλάτηση πιθανών κρουσμάτων, μια διαδικασία που χωρίς την χρήση νέων τεχνολογιών είναι αρκετά χρονοβόρα και αναξιόπιστη καθώς κάποιο άτομο μπορεί να μην μπορεί να μην γνωρίζει πολλούς από τους ανθρώπους που ήρθε σε επαφή και πιθανώς να μετέδωσε τον ιό. (Google, 2020).

Αφού κάποιος χρήστης κατεβάσει την εφαρμογή πρέπει να ενεργοποιήσει τις ειδοποιήσεις αυτές από τις ρυθμίσεις του συστήματός του. Με την ενεργοποίηση των ειδοποιήσεων, στο τηλέφωνο του χρήστη δημιουργείται ένα τυχαίο ID το οποίο αλλάζει κάθε 10 με 20 λεπτά για να διασφαλιστεί το απόρρητο στα δεδομένα του χρήστη (ταυτοποίηση, τοποθεσία κ.α.). Όσα τηλέφωνα έχουν εγκατεστημένη εφαρμογή και οι χρήστες έχουν ενεργοποιήσει αυτές τις ειδοποιήσεις, ανταλλάζουν αυτούς τους τυχαίους αριθμούς όταν βρεθούν σε κοντινή απόσταση. Αν κάποιος χρήστης δηλώσει ότι βρέθηκε θετικός στον covid-19 τότε τα τηλέφωνα των χρηστών που έχουν λάβει το ID από τη συσκευή του τις τελευταίες μέρες λαμβάνουν ειδοποίηση για πιθανή μόλυνση και οδηγίες προστασίας για αυτούς και τους συμπολίτες τους. (Google, 2020).

Σύμφωνα με την Google και την Apple, οι δύο εταιρίες δεν καταγράφουν κανένα προσωπικό δεδομένο σχετικά με την ταυτότητα και την τοποθεσία των χρηστών. (Google, 2020).

Ο τρόπος με τον οποίο λειτουργεί το σύστημα ειδοποιήσεων περιγράφεται στην παρακάτω εικόνα.



Εικόνα 1. Αναπαράσταση λειτουργίας του συστήματος ειδοποιήσεων Apple – Google (Google, 2020)

Σύμφωνα με το xda-developers αρκετά κράτη έχουν αναπτύξει τέτοιες εφαρμογές ή έχουν εγκριθεί από την Apple και Google για την ανάπτυξη των εφαρμογών αυτών. Καθώς δεν υπάρχουν δεδομένα για την ανάπτυξη τέτοιων εφαρμογών σε παγκόσμιο επίπεδο, οι εφαρμογές αυτές βρέθηκαν από μία κρυφή λίστα της Google με τα package names εφαρμογών που έχουν δικαίωμα να χρησιμοποιούν το συγκεκριμένο API.

Region	App Name	Android Package Name
Austria	Stopp Corona	at.rotekreuz.stopcorona
Belgium	Coronalert	be.sciensano.coronalert
Brazil	Coronavirus – SUS	br.gov.datasus.guardioes
Canada	COVID Alert	ca.gc.hcsc.canada.stopcovid
Croatia	Stop COVID-19	hr.miz.evidencijakontakata
Cyprus	CovTracer-EN	cy.gov.dmrld.covtracer
Czech Republic	eRouška	cz.covid19cz.erouska
Denmark	Smittestop	com.netcompany.smittestop_exposure_notification
Ecuador	ASI	ec.gob.asi.android
Estonia	Hoia	ee.tehik.hoia
Finland	Koronavilkku	fi.thl.koronahaavi
Germany	Corona-Warn-App	de.rki.coronawarnapp
Gibraltar	Beat Covid Gibraltar	com.gha.covid.tracker
Greece	Exo	com.pathcheck.gr.bt
Ireland	Covid Tracker	com.covidtracker.hse
Italy	Immuni	it.ministerodellasalute.immuni
Japan	COCOA – COVID-19 Contact App	jp.go.mhlw.covid19radar
Kazakhstan	Saqbol	kz.nitec.bizbirgemiz
Latvia	Apturi Covid Latvia	lv.spkc.gov.apuricovid
Lithuania	Korona Stop LT	lt.nvsc.coronawarnapp
Malta	COVID Alert Malta	mt.gov.dp3t
Netherlands	CoronaMelder	nl.rijksoverheid.en
New Zealand	NZ COVID Tracer	nz.govt.health.covidtracer
Northern Ireland	StopCOVID NI	net.hscni.covidtracker
Norway	Smittestopp	no.fhi.smittestopp_exposure_notification
Panamá	Protégeate Panamá	pa.gob.protegete
Poland	ProteGO Safe	pl.gov.mc.protegosafe
Portugal	STAYAWAY COVID	fct.inesctec.stayaway
Russia	Госуслуги.COVID трекеп	com.minsvyaz.gosuslugi.exposurenotificationdroid
Saudi Arabia	Tabaud	sa.gov.nic.tabaud
Scotland	Protect Scotland	gov.scot.covidtracker

Slovenia	OstaniZdrav	si.gov.ostanizdrav
South Africa	COVIDConnect	za.gov.health.covidconnect
Spain	Radar COVID	es.gob.radarcovid
Switzerland	SwissCovid	ch.admin.bag.dp3t
United Kingdom	NHS COVID-19	uk.nhs.covid19.production
United Kingdom – Bermuda	WeHealth Bermuda	org.wehealth.exposure
United Kingdom – Jersey	Jersey COVID Alert	com.governmentofjersey.jerseycovidalert
Uruguay	Coronavirus UY	uy.gub.salud.plancovid19uy
USA – Alabama	GuideSafe	gov.adph.exposurenotifications
USA – Arizona	Covid Watch	gov.azdhs.covidwatch.android
USA – California	California COVID Notify	gov.ca.covid19.exposurenotifications
USA – Colorado	CO Exposure Notifications	gov.co.cdphe.exposurenotifications
USA – Connecticut	COVID Alert CT	gov.ct.covid19.exposurenotifications
USA – Delaware	Covid Alert DE	gov.de.covidtracker
USA – District of Columbia	DC CAN	gov.dc.covid19.exposurenotifications
USA – Guam	Guam Covid Alert	org.pathcheck.guam.bt
USA – Hawaii	AlohaSafe Alert	org.alohasafe.alert
USA – Louisiana	COVID Defense	org.pathcheck.la.bt
USA – Maryland	MD COVID Alert	gov.md.covid19.exposurenotifications
USA – Michigan	MI COVID Alert	gov.michigan.MiCovidExposure
USA – Minnesota	COVIDaware MN	org.pathcheck.covidsafepathsBt.mn
USA – Nevada	Covid Trace Nevada	gov.nv.dhhs.en
USA – New Jersey	COVID Alert NJ	com.nj.gov.covidalert
USA – New York	COVID Alert NY	gov.ny.health.proximity
USA – North Carolina	SlowCOVIDNC	gov.nc.dhhs.exposurenotification
USA – Oregon	Oregon Exposure Notifications	gov.or.covid19.exposurenotifications
USA – Pennsylvania	COVID Alert PA	gov.pa.covidtracker
USA – Puerto Rico	Rastrea el Virus BT	org.pathcheck.pr.bt
USA – South Carolina	South Carolina Safer Together	musc.exposurenotification

USA – North Dakota & Wyoming	Care19 Alert	com.proudcrowd.exposure
USA – Utah	UT Exposure Notifications	gov.ut.covid19.exposurenotifications
USA – Virginia	COVIDWise	gov.vdh.exposurenotification
USA – Washington	Washington	gov.wa.doh.exposurenotifications
USA – Wisconsin	Wisconsin	gov.wi.covid19.exposurenotifications

Πίνακας 2 Κράτη που έχουν δημιουργήσει εφαρμογές βασισμένες στο σύστημα ειδοποιήσεων της Apple - Google (xda developers, 2021)

Στον παραπάνω πίνακα βρίσκετε και η Ελλάδα με την εφαρμογή Exo ωστόσο, η εφαρμογή δεν είναι διαθέσιμη και δεν υπάρχει καμία πληροφορία γι' αυτή.

1.3. Εφαρμογές ιχνηλάτησης και προσωπικά δεδομένα

Παρά την προσπάθεια των δύο τεχνολογικών κολοσσών (Apple & Google) και παρ' όλη την προσπάθεια των κρατών να αναπτύξουν τέτοιες εφαρμογές, η υιοθέτησή τους από τον κόσμο δεν είναι η αναμενόμενη. Η αντίστοιχη εφαρμογή στην Αυστραλία που κυκλοφορεί από τον Μάιο του 2020 έχει κατέβει μόνο σε 7.1 εκ. συσκευές, ενώ ο στόχος που είχε τεθεί ήταν να ξεπεράσει το 40% του πληθυσμού (26 εκ.). Η ίδια κατάσταση παρατηρείται και σε άλλες χώρες, ενώ στη Σιγκαπούρη η χρήση της εφαρμογής έχει επιβληθεί δια νόμου από τον Οκτώβριο του 2020. (Chan & Saqib, 2021).

Ο λόγος της αποτυχίας των εφαρμογών αυτών όσο αφορά την υιοθέτησή τους από τον γενικό πληθυσμό, είναι η ασφάλεια των δεδομένων που συλλέγονται από αυτές. Οι εφαρμογές αυτές συλλέγουν ευαίσθητα δεδομένα όπως η τοποθεσία και ιατρικές πληροφορίες για τον χρήστη. Παρ όλες τις διαβεβαιώσεις ότι οι πληροφορίες αυτές είναι ανώνυμες, ο χρήστης μπορεί να ταυτοποιηθεί πλήρως, όταν τα δεδομένα που συλλέγει η εφαρμογή διασταυρωθούν με δεδομένα από άλλες πηγές. Έχουμε λοιπόν ψευδό-ανωνυμία και όχι ανωνυμία. Τα δεδομένα αυτά μπορούν να αποκτηθούν εύκολα από τρίτους αξιοποιώντας γνωστές ευπάθειες του συστήματος Bluetooth του οποίου η ασφάλεια εξαρτάται από την εκάστοτε συσκευή. Πολλά ερωτηματικά υπάρχουν και σχετικά με την ασφάλεια αποθήκευσης των δεδομένων αυτών από τα κράτη. (Cecannecchia, Cioffi & Lugi, 2020).

Παραδείγματα καταπάτησης της ιδιωτικότητας των πολιτών μέσω εφαρμογών ιχνηλάτησης εντοπίστηκαν στο Bahrain, Kuwait και Νορβηγία στα μέσα του 2020. Η Νορβηγία σταμάτησε

τη λειτουργία της εφαρμογής μετά από παρέμβαση της διεθνούς αμνηστίας, καθώς σε έλεγχο που έγινε διαπιστώθηκε ότι η εφαρμογή ήταν αρκετά ευάλωτη σε επιθέσεις από τρίτους με κίνδυνο τη διαρροή των προσωπικών στοιχείων των χρηστών. Η διεθνής αμνηστία παρότρυνε τις κυβερνήσεις του Bahrein και Kuwait να σταματήσουν τη λειτουργία των εφαρμογών τους καθώς έστελναν την τοποθεσία των χρηστών σε πραγματικό χρόνο σε κυβερνητικές βάσεις δεδομένων. (Amnesty international, 2020).

Μια μεγάλη αδυναμία ασφαλείας στην εφαρμογή EHTERAZ του Qatar οδήγησε στην δημοσίευση ευαίσθητων προσωπικών δεδομένων για πάνω από 1 εκατομμύριο χρήστες, ενώ η εφαρμογή είχε γίνει υποχρεωτική από τις 22 Μάϊου 2020. Στο Bahrein πρόσβαση στα δεδομένα της εφαρμογής είχε ένα εθνικό τηλεοπτικό show με την ονομασία “Are you Home?”. Το τηλεοπτικό show επιβράβευε χρήστες που ήταν στο σπίτι παίρνοντας τους τηλέφωνο. Ενώ η συμμετοχή των χρηστών στο show αρχικά ήταν υποχρεωτική, η κυβέρνηση του Bahrein αργότερα έδωσε τη δυνατότητα στους χρήστες να επιλέξουν αν θα συνεχίσουν αν συμμετέχουν ή όχι. (Amnesty international, 2020).

Δεν είναι όμως μόνο οι παραπάνω περιπτώσεις, πολλά κράτη αποθηκεύουν τα δεδομένα σε κρατικές υποδομές χωρίς να κάνουν γνωστά στους πολίτες τα πρωτόκολλα ασφαλείας που χρησιμοποιούν καθώς επίσης δεν δίνουν καμία εγγύηση ότι τα δεδομένα αυτά δεν θα χρησιμοποιηθούν και για άλλους σκοπούς ή δεν θα δοθούν – πουληθούν σε τρίτους. (Amnesty international, 2020).

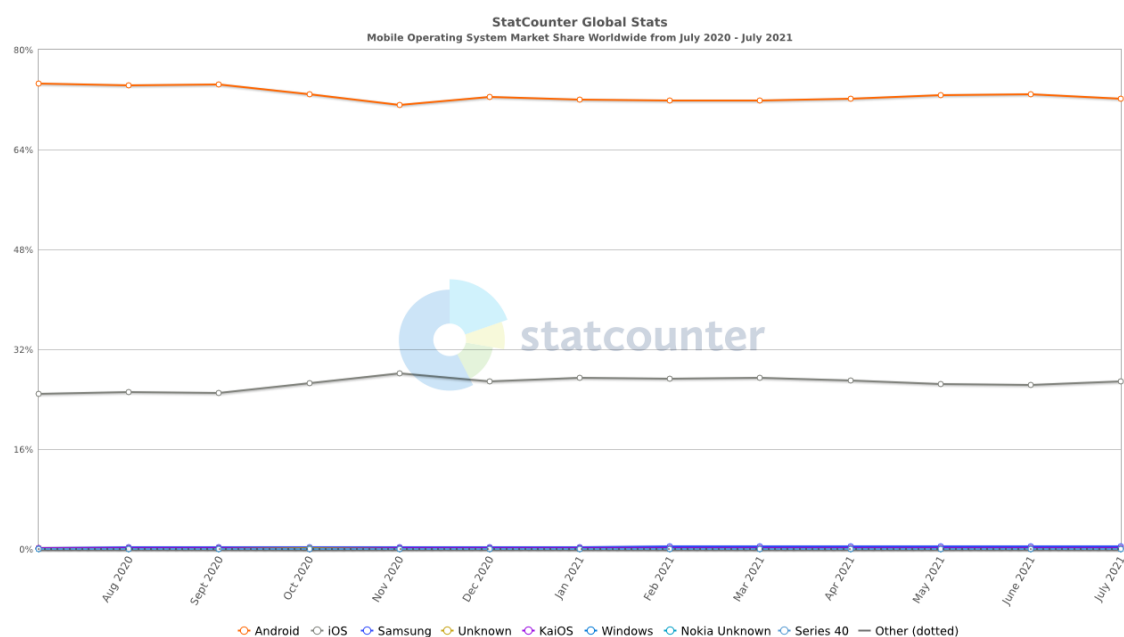
2. Δημιουργία ενημερωτικής εφαρμογής για τον covid-19

Στόχος της συγκεκριμένης εργασίας είναι η δημιουργία ενημερωτικής εφαρμογής με σεβασμό στα προσωπικά δεδομένα του χρήστη. Με τη χρήση της εφαρμογής ο χρήστης θα μπορεί να ενημερώνεται για την πορεία της πανδημίας στη χώρα, και θα μπορεί να απαντά σε σχετικές έρευνες σχετικά με την πανδημία. Στόχος είναι η δημιουργία ενός ερευνητικού εργαλείου όπου θα αναλύονται οι τάσεις στον γενικό πληθυσμό σε θέματα της πανδημίας.

Η εφαρμογή χωρίζεται σε δύο κομμάτια, την mobile εφαρμογή και ένα web app όπου κάθε ενδιαφερόμενος μπορεί να δει τα αποτελέσματα των ερευνών με τη μορφή διαγραμμάτων. Στο web app η επιστημονική ομάδα μπορεί να δημιουργεί και να διαχειρίζεται τις έρευνες υπό τη μορφή ερωτηματολογίων.

2.1. Επιλογή τεχνολογιών

Στόχος είναι η εφαρμογή να τρέχει σε όσο το δυνατόν περισσότερες mobile συσκευές. Για τον λόγο αυτό επιλέχθηκε το λειτουργικό android της Google για την αρχική ανάπτυξη. Σύμφωνα με το statcounter.com το λειτουργικό android κατέχει περίπου το 72% της παγκόσμιας αγοράς για την περίοδο Ιούλιος 2020 – Ιούλιος 2021 σε σχέση με το λειτουργικό IOS της Apple.



Εικόνα 2: Mobile Operating System Market Share Worldwide (statcounter, 2021)

Για την ανάπτυξη της mobile εφαρμογής επιλέχθηκε το android studio καθώς είναι το επίσημο εργαλείο για την ανάπτυξη εφαρμογών σε περιβάλλον android, ενώ επιλέχθηκε η γλώσσα προγραμματισμού KOTLIN έναντι της JAVA. Η επιλογή έγινε καθώς η γλώσσα KOTLIN είναι πολύ νεότερη γλώσσα και έχει αρκετά νέα στοιχεία τα οποία βοηθούν στην παραγωγή γρήγορου και ασφαλέστερου κώδικα σε σχέση με την απαρχαιωμένη JAVA. Διαχειρίζεται πολύ καλύτερα ασύγχρονες κλήσεις δικτύου και είναι 100% συμβατή με τη JAVA, που σημαίνει ότι ένα project μπορεί να είναι γραμμένο και με τις δύο γλώσσες χωρίς κανένα πρόβλημα. (Android developers, 2021)

Για τη δημιουργία του web app, επιλέχθηκε η γλώσσα php και το Laravel framework με βάση δεδομένων MySQL. Η επιλογή του Laravel framework έγινε αρχικά λόγω της μεγάλης κοινότητας που το υποστηρίζει, του κατατοπιστικότατου documentation για την κάθε έκδοση και των πολλών ενσωματωμένων λειτουργιών.

2.2. Πηγές δεδομένων

Όπως περιγράψαμε προηγουμένως στόχος της εφαρμογής είναι η ενημέρωση του χρήστη σχετικά με την πορεία της πανδημίας στην Ελλάδα και η παρότρυνση αυτού στη συμμετοχή ερευνών με τη χρήση ερωτηματολογίων. Τα στοιχεία για την πορεία της πανδημίας στην Ελλάδα τα βρίσκει κανείς στην επίσημη σελίδα του ΕΟΔΥ (Εθνικός Οργανισμός Δημόσιας Υγείας), ωστόσο ο οργανισμός δεν παρέχει κάποιο API όπου να μπορεί κάποιος να τραβήξει τα στοιχεία που θέλει. Η εθελοντική ομάδα [COVID-19 RESPONSE GREECE](#), έχει αναπτύξει ένα τέτοιο API το οποίο είναι διαθέσιμο σε όλους μέσω GitHub, [covid19-greece-api](#). Τα δεδομένα που χρησιμοποιούν προέρχονται από τον ΕΟΔΥ και από το ίδρυμα JOHNS HOPKINS. Η κλήσεις στο API γίνονται μέσω HTTPS και τα δεδομένα επιστέφονται με τη μορφή JSON στον χρήστη.

Το API αυτό είναι και η μοναδική πηγή δεδομένων της εφαρμογής για την πορεία της πανδημίας. Τα υπόλοιπα δεδομένα της εφαρμογής προέρχονται από το web app.

2.3. Λογότυπο και όνομα εφαρμογής

Το όνομα και το λογότυπο της εφαρμογής είναι σημαντικά και πρέπει να αντιπροσωπεύουν την λειτουργία αυτής. Όπως είπαμε και προηγουμένως στόχος της εφαρμογής είναι η ενημέρωση των χρηστών ως προς την πορεία της πανδημίας στην Ελλάδα καθώς και η συμμετοχή αυτών σε έρευνες με τη μορφή ερωτηματολογίου. Το μήνυμα που θέλουμε να περάσουμε είναι ότι μόνο με τη σωστή ενημέρωση και γνώση των κανόνων προστασίας θα μπορέσουμε να κερδίσουμε τη μάχη με τον ιό. Έτσι λοιπόν το λογότυπο και το όνομα πρέπει να συμβολίζουν όλα τα παραπάνω.

Με βάση τα παραπάνω δημιουργήθηκαν δύο λογότυπα.



Εικόνα 3: Λογότυπο πρόταση 1



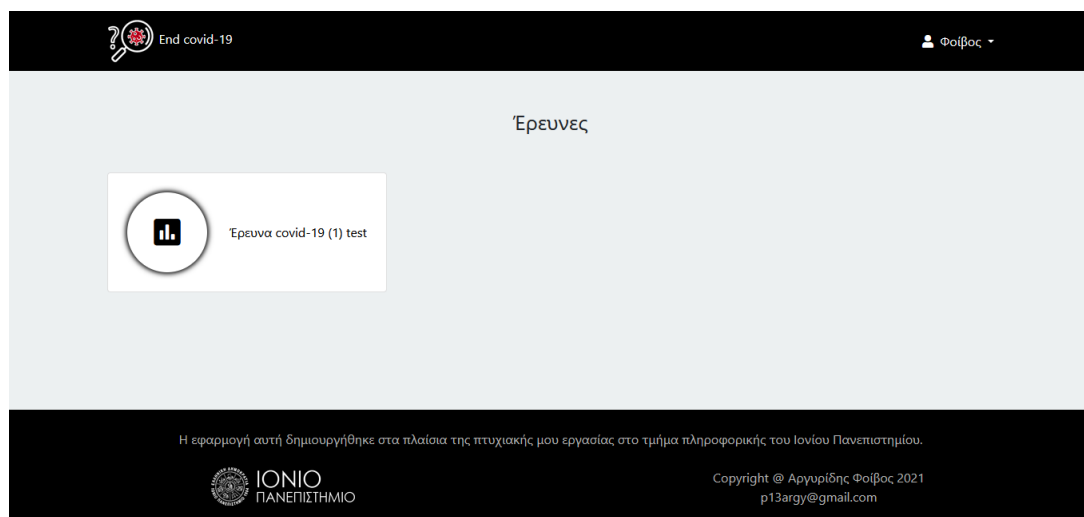
Εικόνα 4: Λογότυπο πρόταση 2

Το δεύτερο λογότυπο αντιπροσωπεύει περισσότερο τους στόχους της εφαρμογής, καθώς ο μεγεθυντικός φακός τοποθετεί τον κορονοϊό στα πλαίσια της έρευνας με το ερωτηματικό να δηλώνει ότι κάτι ψάχνουμε, ενώ συσχετίζεται άμεσα και με τα ερωτηματολόγια. Το πρώτο λογότυπο συμβολίζει απλά μια ασπίδα προστασίας (γνώση) ενάντια στον κορονοϊό.

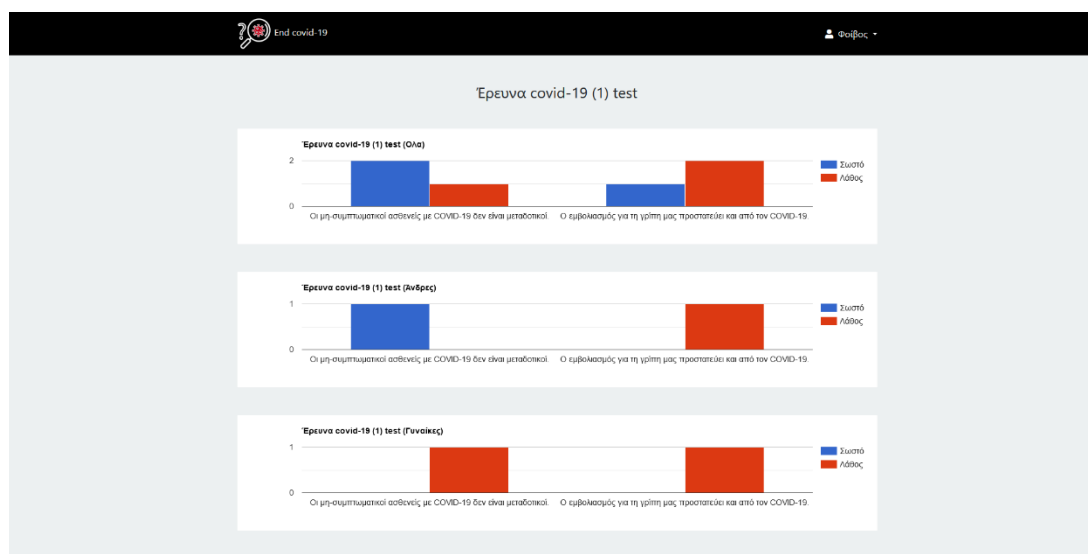
Σαν όνομα της εφαρμογής επιλέχθηκε το End covid-19, το οποίο μαζί με το λογότυπο καλύπτουν πλήρως τις απαιτήσεις που θέσαμε προηγουμένως.

2.4. Περιγραφή λειτουργίας web app

Το web app απευθύνεται σε δύο τύπους χρηστών, τους κοινούς χρήστες και την ερευνητική ομάδα. Όλοι οι χρήστες στην αρχική οθόνη βλέπουν τις διαθέσιμες έρευνες και κάνοντας κλικ πάνω σε αυτές βλέπουν τις απαντήσεις των χρηστών με τη μορφή διαγραμμάτων.



Εικόνα 5: Αρχική εικόνα εφαρμογής.



Εικόνα 6: Διαγραμματική απεικόνιση αποτελεσμάτων έρευνας

Ο χρήστης βλέπει τα αποτελέσματα γενικά και ανά φύλο.

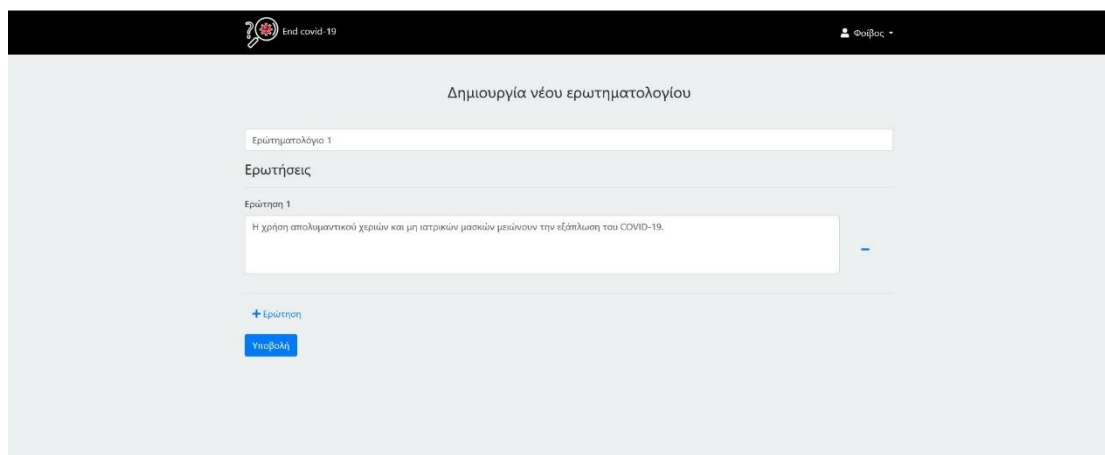
Όσο αφορά τη ερευνητική ομάδα οι χρήστες αυτής έχουν τη δυνατότητα να κάνουν login ώστε να μπορούν να δημιουργήσουν και να διαχειριστούν ερωτηματολόγια.

Εικόνα 7: Login εφαρμογής

Εδώ υπάρχουν τρεις κατηγορίες χρηστών με διαφορετικά δικαιώματα, ο Super admin, οι admins και οι editors. Οι admins και ο Super admin έχουν τη δυνατότητα να βλέπουν να επεξεργάζονται και να δημοσιεύουν τα ερωτηματολόγια όλων των χρηστών, να δημιουργούν νέους χρήστες και να επεξεργάζονται τους ρόλους αυτών. Οι admins μπορούν να αλλάξουν έναν ρόλο μόνο από editor σε admin και όχι το αντίστροφο. Μόνο ο Super admin μπορεί να μετατρέψει έναν admin σε editor. Μόνο οι admins και ο Super admin μπορούν να διαγράψουν χρήστες. Σε κάθε διαγραφή χρήστη τα ερωτηματολόγια που έχει δημιουργήσει ο συγκεκριμένος χρήστης περνάνε στην κυριότητα του Super admin.

Εικόνα 8: Διαγραφή χρήστη.

Όπως είπαμε και παραπάνω όλοι οι χρήστες μπορούν να δημιουργούν ερωτηματολόγια. Η δημιουργία των ερωτηματολογίων είναι μια πάρα πολύ εύκολη διαδικασία και ο χρήστης μπορεί να προσθέσει όσες ερωτήσεις επιθυμεί. Πατώντας το κουμπί νέα ερώτηση προστίθεται νέο πεδίο για να προσθέσει ο χρήστης την ερώτηση του. Με το κουμπάκι – (πλην) ο χρήστης μπορεί να αφαιρέσει κάποια ερώτηση. Οι ερωτήσεις προς το παρόν είναι του τύπου ΣΩΣΤΟ – ΛΑΘΟΣ, οπότε δεν χρειάζεται ο χρήστης να προσθέσει πιθανές απαντήσεις. Για να είναι ορατό το ερωτηματολόγιο στις εφαρμογές, πρέπει αυτό να είναι ενεργό και μόνο οι admin και ο Super admin μπορούν να το ενεργοποιήσουν, ενώ μόνο ένα ερωτηματολόγιο μπορεί να είναι ενεργό κάθε φορά. Με την ενεργοποίηση του το ερωτηματολόγιο δεν μπορεί πλέον να υποστεί καμία επεξεργασία, ενώ με τη διαγραφή του διαγράφονται και όλες οι απαντήσεις του.

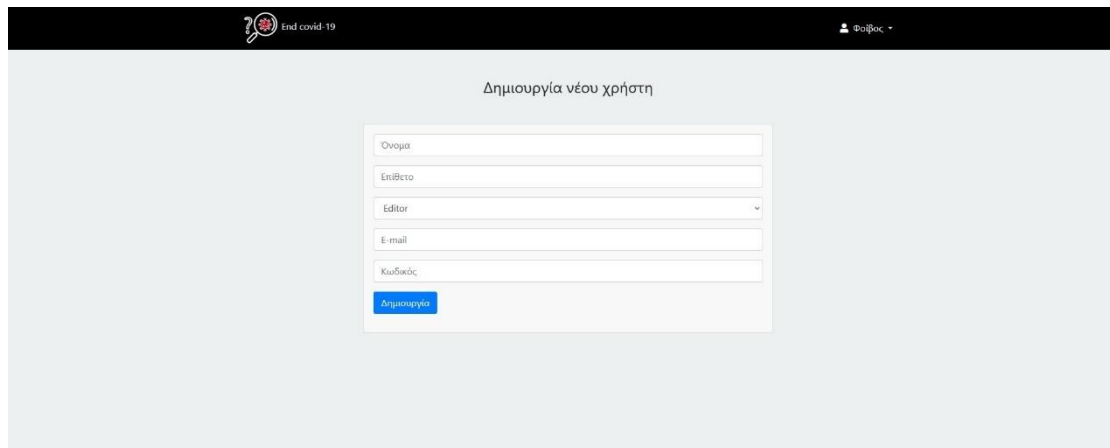


Εικόνα 9: Δημιουργία νέου ερωτηματολογίου.

Για να είναι τα αποτελέσματα ενός ερωτηματολογίου ορατά από τους χρήστες του web app, το ερωτηματολόγιο θα πρέπει να είναι ορατό, και όπως είπαμε και πιο πάνω μόνο οι Admins και ο Super admin μπορούν να το κάνουν αυτό. Κάνοντας ένα ερωτηματολόγιο ορατό και το αντίστροφο, εμφανίζεται σχετικό μήνυμα στον χρήστη ότι το συγκεκριμένο ερωτηματολόγιο είναι πλέον ορατό ή το αντίθετο.

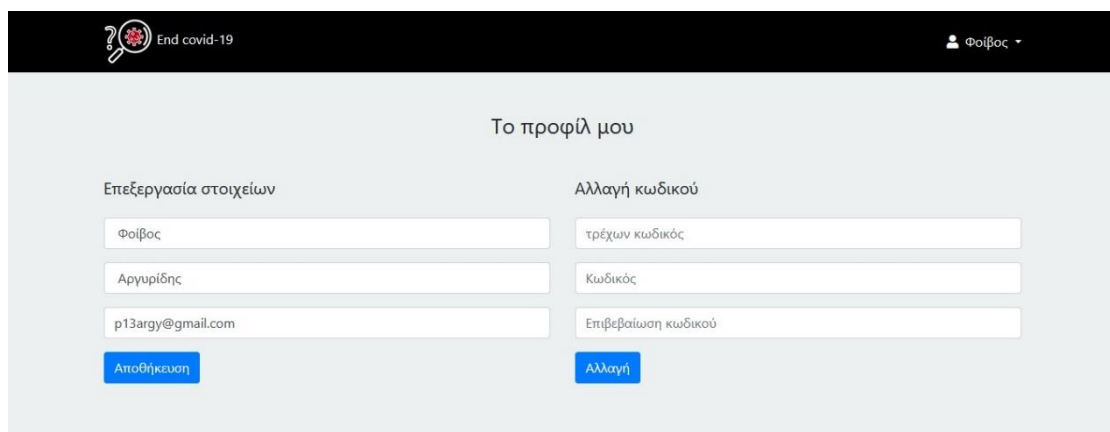
Οι admins και ο Super admin έχουν το δικαίωμα να δημιουργούν χρήστες. Για τη δημιουργία ενός νέου χρήστη είναι απαραίτητα τα παρακάτω πεδία:

- Όνομα
- Επίθετο
- Ρόλος (editor – Admin)
- E-mail
- Κωδικός



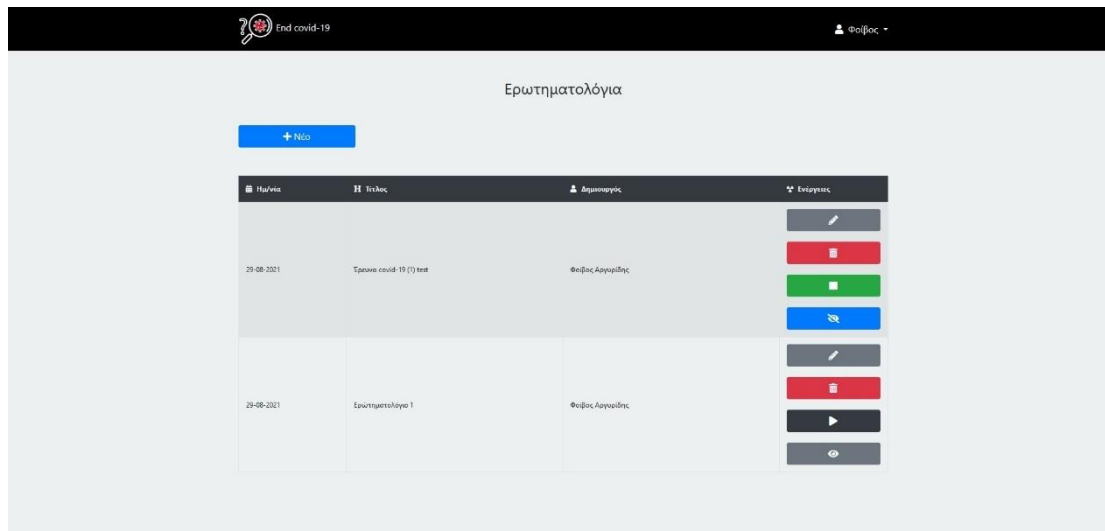
Εικόνα 10: Δημιουργία νέου χρήστη

Όλοι οι χρήστες μπορούν να επεξεργαστούν τα στοιχεία τους και να αλλάξουν των κωδικό τους από το προφίλ. Για την αλλαγή του κωδικού απαιτείται ο τρέχων κωδικός και η επιβεβαίωση του νέου κωδικού.



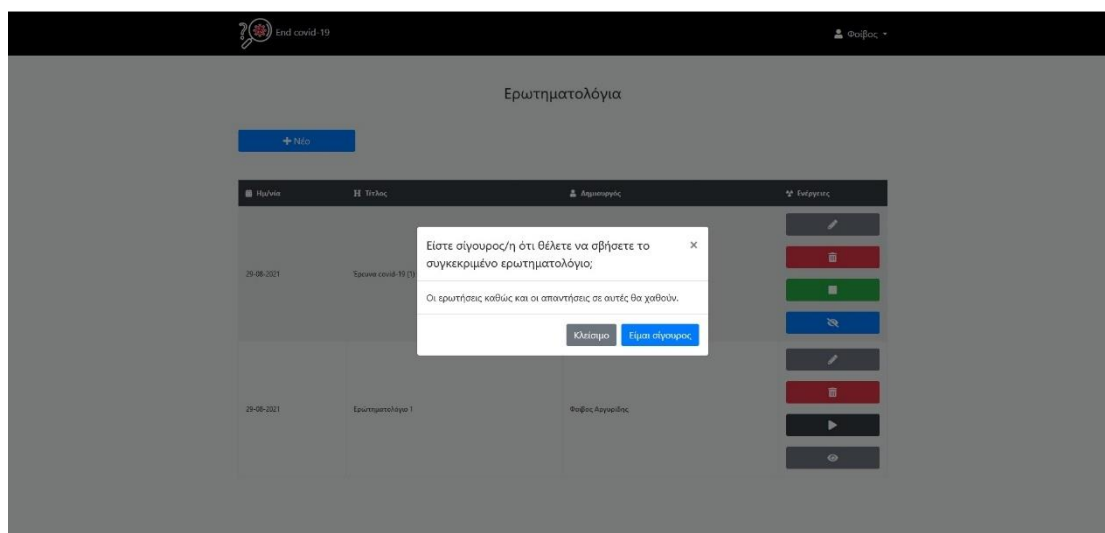
Εικόνα 11: Προφίλ χρήστη

Για να αλλάξουν επιτυχώς τα στοιχεία του χρήστη δεν πρέπει το νέο email (εφόσον αλλάξει), να υπάρχει ήδη σε κάποιον άλλον χρήστη.



Εικόνα 12: Διαχείριση ερωτηματολογίων (Admins - Super admin)

Στην παραπάνω εικόνα βλέπουμε ότι κάθε ερωτηματολόγιο έχει 4 κουμπάκια. Το πρώτο κουμπάκι με το μολύβι είναι για να επεξεργαστούμε το ερωτηματολόγιο, όπου ανοίγει μία νέα σελίδα παρόμοια με αυτή της δημιουργίας του ερωτηματολογίου με όλες τις ερωτήσεις ώστε να τις επεξεργαστούμε. Το δεύτερο κουμπάκι διαγράφει το ερωτηματολόγιο. Πριν προχωρήσει όμως στη διαγραφή ο χρήστης ενημερώνεται ότι θα διαγραφούν και όλες οι απαντήσεις σε αυτό. Το τρίτο κουμπάκι ενεργοποιεί και απενεργοποιεί το ερωτηματολόγιο. Το κουμπί αλλάζει ανάλογα με την κατάσταση του ερωτηματολογίου. Τέλος το τελευταίο κουμπί κάνει ορατό το ερωτηματολόγιο ώστε να μπορούν να εμφανίζονται τα αποτελέσματα. Πάλι εδώ το κουμπί αλλάζει ανάλογα με την κατάσταση του ερωτηματολογίου.



Εικόνα 13: Διαγραφή ερωτηματολογίου.

2.5. Database schema

Για την υλοποίηση της παραπάνω εφαρμογής είναι απαραίτητοι οι παρακάτω πίνακες με τις συσχετίσεις τους.

- **users**

Οι χρήστες της εφαρμογής. Κάθε χρήστης μπορεί να έχει μόνο έναν ρόλο. Κάθε χρήστης μπορεί να έχει πολλά ερωτηματολόγια.

id	First_name	Last_name	email	Email_verified_at	password	Role_id	Remember_token	created_at	updated_at
----	------------	-----------	-------	-------------------	----------	---------	----------------	------------	------------

Εδώ τα πεδία Email_verified_at και Remember token είναι για την επιβεβαίωση του mail του χρήστη και την ανάκτηση του κωδικού του σε περίπτωση που τον ξέχασε. Και στις δύο περιπτώσεις δεν έχουν δημιουργηθεί ακόμα οι συγκεκριμένες διαδικασίες.

- **roles**

Οι ρόλοι των χρηστών. Κάθε ρόλος μπορεί να αντιστοιχεί σε πολλούς χρήστες.

id	name	created_at	updated_at
----	------	------------	------------

- **questionnaires**

Τα ερωτηματολόγια. Κάθε ερωτηματολόγιο μπορεί να έχει πολλές απαντήσεις ενώ μόνο έναν χρήστη.

id	user_id	title	questions	editable	display	active	created_at	updated_at
----	---------	-------	-----------	----------	---------	--------	------------	------------

Οι ερωτήσεις αποθηκεύονται ως string array, το πεδίο editable γίνεται true την πρώτη φορά που το πεδίο active γίνεται true. Και υπάρχει για να μην μπορεί κάποιος χρήστης να επεξεργαστεί το ερωτηματολόγιο εφόσον έχει ενεργοποιηθεί έστω και μία φορά, ενώ το πεδίο display είναι για την παρουσίαση των αποτελεσμάτων στους χρήστες.

- **answers**

Οι απαντήσεις των ερωτηματολογίων. Κάθε απάντηση μπορεί να ανήκει μόνο σε ένα ερωτηματολόγιο, μόνο σε ένα φύλο μόνο σε μία ηλικιακή ομάδα και μόνο σε έναν χρήστη (ως χρήστης ορίζεται ο χρήστης της mobile εφαρμογής και είναι ανώνυμος, κρατάμε το token της εφαρμογής το οποίο παράγεται από το firebase cloud messaging). Ορίζοντας ως μοναδικό κλειδί το questionnaire_id και το token_id

εξασφαλίζουμε ότι κάθε χρήστης μπορεί να απαντήσει μόνο μία φορά ένα ερωτηματολόγιο.

id	gender_id	age_group_id	questionnaire_id	token_id	answers	created_at	Updated_at
----	-----------	--------------	------------------	----------	---------	------------	------------

Οι απαντήσεις αποθηκεύονται ως string array όπως και στις ερωτήσεις.

- **firebase_tokens**

Τα tokens που παράγονται από το firebase cloud messaging και είναι μοναδικά για κάθε συσκευή που τρέχει το mobile application. Κάθε token μπορεί να ανήκει σε πολλές απαντήσεις.

id	token	created_at	updated_at
----	-------	------------	------------

- **genders**

Προσδιορισμός του σεξουαλικού φύλου του χρήστη. Κάθε gender μπορεί να ανήκει σε πολλές απαντήσεις.

id	name	created_at	updated_at
----	------	------------	------------

- **age_groups**

Οι ηλικιακές ομάδες των χρηστών της εφαρμογής. Κάθε age_group μπορεί να ανήκει σε πολλές απαντήσεις.

id	group	created_at	updated_at
----	-------	------------	------------

Η Laravel με το migration δημιουργεί και άλλους πίνακες μαζί με τον πίνακα **password_resets** όπου το σύστημα δημιουργεί ένα μοναδικό token για το mail του χρήστη το οποίο χρησιμοποιείται για την επαναφορά του κωδικού του σε περίπτωση που το έχει ξεχάσει. Η διαδικασία αυτή δεν έχει υλοποιηθεί προς το παρόν καθώς απαιτεί σύνδεση με κάποιον mail server ώστε να φεύγουν αυτοματοποιημένα mail. Οι πίνακες **migrations** και **failed_jobs** αφορούν τη Laravel αυτή καθ' αυτή.

2.6. Code structure

Ο κώδικας της εφαρμογής χωρίζεται σε διάφορα κομμάτια, θα αναλύσουμε το folder structure και τα functions που δημιουργήθηκαν σε κάθε class. Η Laravel ακολουθεί το μοτίβο Model – View – Controller.

- Οι controllers επεξεργάζονται τα requests των χρηστών και επιστέφουν δεδομένα στα views χρησιμοποιώντας τα μοντέλα.
- Τα models αλληλοεπιδρούν με τη βάση δεδομένων.
- Τα views σερβίρουν τα αποτελέσματα στον χρήστη.

2.6.1. Models

Δημιουργήθηκαν τα εξής models:

- **User**

Περιέχει τα εξής functions τα οποία καθορίζουν τις σχέσεις με τα άλλα models χρησιμοποιώντας τα expressions `hasOne()`, `hasMany()`, `belongsTo()`, `belongsToMany()` της Laravel.

- **public function questionnaire()**

Κάθε χρήστης μπορεί να έχει πολλά ερωτηματολόγια.

- **public function role()**

Κάθε χρήστης ανήκει μόνο σε έναν ρόλο.

- **public function hasAdminRole()**

Returns true αν ο χρήστης ανήκει σε κάποιον από τους ρόλους Admin, Super admin και false αν όχι.

Επίσης η Laravel δίνει τη δυνατότητα προστασίας κάποιων πεδίων στη βάση δεδομένων μέσω του **protected \$hidden array[]**. Τα πεδία αυτά δεν εμφανίζονται στον χρήστη. Στο array αυτό βρίσκονται τα πεδία password και remember_token. Επίσης η Laravel παρέχει και τη δυνατότητα αποφυγής mass assignment από κακόβουλους χρήστες μέσω του **protected \$fillable array[]**. Στο array αυτό βρίσκονται τα πεδία email, password και role_id.

- **Role**

Περιέχει μόνο το relationship function με το model User.

- **public function users()**

Κάθε ρόλος έχει πολλούς χρήστες.

- **Questionnaire**

Περιέχει τις σχέσεις μεταξύ User και Answer model

- **public function user()**
Κάθε ερωτηματολόγιο ανήκει σε έναν και μόνο χρήστη.
- **public function answers()**
Κάθε ερωτηματολόγιο έχει πολλές απαντήσεις

Πάλι η Laravel μας δίνει τη δυνατότητα να κάνουμε cast κάποιο πεδίο σε ότι type θέλουμε μέσω του protected \$casts array[], έτσι το array αυτό περιέχει το πεδίο questions το οποίο θέλουμε να είναι array.

- **Gender**

Περιέχει τη σχέση με το model Answer

- **public function answers()**
Κάθε φύλο έχει πολλές απαντήσεις

- **FirebaseToken**

Περιέχει τη σχέση με το model Answer

- **public function answers()**
Κάθε token έχει πολλές απαντήσεις.

Επίσης χρησιμοποιούμε την protected \$fillable array, για να προστατέψουμε το token από mass assignment και την protected \$hidden array, για να κρύψουμε το token από τους χρήστες.

- **Answer**

Περιέχει τις σχέσεις με τα models Questionnaire, FirebaseToken, Gender, AgeGroup.

- **public function questionnaire()**
Κάθε απάντηση ανήκει μόνο σε ένα ερωτηματολόγιο.
- **public function firebaseToken()**
Κάθε απάντηση ανήκει μόνο σε ένα FirebaseToken
- **public function gender()**
Κάθε απάντηση ανήκει μόνο σε ένα φύλο
- **public function age()**
Κάθε απάντηση ανήκει μόνο σε μία ηλικιακή ομάδα

Επίσης χρησιμοποιούμε την protected \$casts array για να κάνουμε cast σε array το πεδίο answers.

- AgeGroup

Περιέχει τη σχέση με το model Answers.

- public function answers()

Κάθε ηλικιακή ομάδα έχει πολλές απαντήσεις.

2.6.2. Controllers

Δημιουργήθηκαν οι εξής Controllers:

- **authController**

Περιέχει τα functions για την ταυτοποίηση του χρήστη με την εφαρμογή.

- **function index()**

Τσεκάρει αν ο χρήστης είναι συνδεδεμένος. Αν είναι τον κάνει redirect στην αρχική σελίδα, αν όχι επιστέφει το view login.

- **function getSign_in()**

Στέλνει τον χρήστη στη σελίδα login μέσω redirect

- **function sign_in(Request \$request)**

Δέχεται ως παράμετρος τα δεδομένα από τη φόρμα login και ταυτοποιεί τον χρήστη. Πριν την ταυτοποίηση ελέγχεται η ορθότητα των στοιχείων του χρήστη (mail, password). Και τα δυο πεδία είναι υποχρεωτικά, ενώ το πεδίο mail πρέπει να είναι της μορφής example@example.com. Για την διαδικασία αυτή χρησιμοποιείται το validate function της Laravel. Αν τα πεδία δεν περάσουν το validation επιστρέφεται σχετικό μήνυμα στον χρήστη. Η διαδικασία αυτή είναι build in στη Laravel. Εάν τα πεδία περάσουν το validation τότε με την build in Auth function της Laravel γίνεται προσπάθεια ταυτοποίησης του χρήστη. Αν τα στοιχεία είναι σωστά τα αποθηκεύουμε σε ένα Session και κάνουμε redirect τον χρήστη στην αρχική σελίδα. Αν τα στοιχεία δεν είναι σωστά τότε επιστέφουμε πίσω στο login view με σχετικό μήνυμα.

- **function logout()**

Κάνουμε logout τον χρήστη και τον επιστέφουμε στην σελίδα login

- **FirestoreController**

Περιέχει το function το οποίο αποθηκεύει το token από τη mobile εφαρμογή στη βάση δεδομένων

- **function postToken(Request \$request)**

Δέχεται το token από την εφαρμογή και το αποθηκεύει στη βάση δεδομένων. Πριν γίνει η αποθήκευση πρέπει να τσεκάρουμε αν όντως έχουμε δεχθεί

token από την εφαρμογή και το token αυτό δεν υπάρχει ήδη στη βάση μας. Εδώ χρησιμοποιούμε διαφορετικό function για το validation καθώς η μέθοδος validate που είδαμε προηγουμένως επιστρέφει τα σφάλματα μαζί με το view από το οποίο ήρθαν τα δεδομένα, θέλουμε όμως απλά να επιστρέψουμε ένα μήνυμα στον χρήστη της εφαρμογής. Για τον λόγο αυτό χρησιμοποιούμε τη μέθοδο Validator::make() που δέχεται παράμετρο το post request του χρήστη της εφαρμογής (δηλαδή το token) και τσεκάρει αν υπάρχει (required) και αν υπάρχει ήδη στη βάση μας (unique:firebase_tokens). Αν το validation πετύχει γίνεται εισαγωγή του token στη βάση και επιστρέφουμε σχετικό μήνυμα στον χρήστη της εφαρμογής, αν όχι επιστρέφουμε πάλι μήνυμα στον χρήστη ότι παρουσιάστηκε κάποιο σφάλμα.

- **homeController**

Περιέχει μια **function index** όπου αρχικά τσεκάρει αν υπάρχει συνδεδεμένος χρήστης και στη συνέχεια βρίσκει όλα τα ορατά ερωτηματολόγια από τη βάση (display = 1 δηλαδή true). Επιστέφει το Home view μαζί με τον συνδεδεμένο χρήστη αν υπάρχει και τα ερωτηματολόγια.

- **questionnaireController**

Περιέχει όλα τα functions για τη δημιουργία, το edit την προβολή και τη διαγραφή των ερωτηματολογίων.

- **function index()**

Βρίσκει τον συνδεδεμένο user και στη συνέχεια τσεκάρει τον ρόλο του. Αν ο ρόλος του user είναι Admin ή Super admin τότε ανακτά όλα τα ερωτηματολόγια από τη βάση δεδομένων με τη βοήθεια της function paginate (για σελιδοποίηση σε περίπτωση που είναι πάνω από 5). Αν ο ρόλος του χρήστη είναι editor τότε γίνεται ανάκτηση μόνο των ερωτηματολογίων που έχει δημιουργήσει ο συγκεκριμένος χρήστης. Τέλος επιστρέφει το view questionnaires μαζί με τον χρήστη και τα ερωτηματολόγια.

- **function getQuestionnaire(\$id)**

Τρέχει τις διαδικασίες για την οπτικοποίηση των απαντήσεων σε διαγράμματα. Δέχεται σαν παράμετρο το id ενός ερωτηματολογίου και αρχικά τσεκάρει για συνδεδεμένο χρήστη και στη συνέχεια βρίσκει το ερωτηματολόγιο που αντιστοιχεί στο id και είναι και ορατό. Αν υπάρχει το ερωτηματολόγιο, τότε αρχικά αποθηκεύονται όλες οι απαντήσεις του

(\$answers_all) και στη συνέχεια οι απαντήσεις ανά φύλο (\$answers_men, \$answers_women, \$answers_other). Οι απαντήσεις πρέπει να επεξεργαστούν και να έρθουν σε μία μορφή ώστε να είναι εύκολο να οπτικοποιηθούν αργότερα από το view. Για τον σκοπό αυτό έχει δημιουργηθεί το function answerData() όπου επιστέφει ένα array με τη μορφή που πρέπει. Για κάθε ομάδα απαντήσεων καλείται η answerData() και τα αποτελέσματα αποθηκεύονται στα αντίστοιχα arrays (\$data_all, \$data_men, \$data_women, \$data_other). Το function επιστρέφει το view questionnaires από το folder public (υπάρχει ίδιο view στο folder users), μαζί με τον ταυτοποιημένο χρήστη (αν υπάρχει), Το ερωτηματολόγιο και όλα τα arrays με τις δομημένες απαντήσεις. Αν δεν βρεθεί ερωτηματολόγιο επιστρέφει την build in 404 σελίδα της Laravel.

○ **function answerData(\$questionnaire, \$answers)**

Ετοιμάζει τα δεδομένα σε κατάλληλη μορφή ώστε να μπορούν να οπτικοποιηθούν στο view με τη βοήθεια διαγραμμάτων. Δέχεται σαν παραμέτρους το ερωτηματολόγιο και τις απαντήσεις του. Αρχικοποιούμε ένα empty array και δηλώνουμε μια Boolean μεταβλητή \$isFirstRun = true. Στη συνέχεια για κάθε απάντηση ορίζουμε την μεταβλητή \$i = 0 και για κάθε απάντηση από το πεδίο answers του πίνακα answers, τσεκάρουμε αν είναι true ή false και αν το isFirstRun είναι true τότε κάνουμε push στο data array που αρχικοποιήσαμε πριν ένα νέο array με τα δεδομένα (πχ , array_push(\$data, array('question' => "", 'true' => 1, 'false' => 0));) Αν το isFistRun είναι false, τότε απλά ανεβάζουμε κατά μία μονάδα την αντίστοιχη τιμή (\$data[\$i]['false']++;). Πριν κλείσει το ενσωματωμένο loop αυξάνουμε την τιμή του i κατά μία μονάδα και πριν κλείσει το αρχικό loop αλλάζουμε την τιμή του isFirstRun σε false; Όπως παρατηρούμε παραπάνω τα πεδία question στο array μας είναι κενά. Πρέπει να πάρουμε τις ερωτήσεις από τον πίνακα question και να κάνουμε update το array, Έτσι λοιπόν ορίζουμε πάλι έναν ακέραιο με τιμή 0 και για κάθε question στον πεδίο questions του πίνακα questionnaires αλλάζουμε το πεδίο question του data array που δημιουργήσαμε προηγουμένως με την ερώτηση. \$data[\$i]['question'] = \$question; Τέλος αυξάνουμε τον ακέραιο κατά μία τιμή. Το function επιστρέφει το data array;

```

array:2 [▼
  0 => array:3 [▼
    "question" => "Οι μη-συμπτωματικοί ασθενείς με COVID-19 δεν είναι μεταδοτικοί."
    "true" => 2
    "false" => 1
  ]
  1 => array:3 [▼
    "question" => "Ο εμβολιασμός για τη γρίπη μας προστατεύει και από τον COVID-19."
    "true" => 1
    "false" => 2
  ]
]

```

Εικόνα 14: Παραδειγμα δομής του data array

- **function addNew()**
Επιστρέφει το view για τη δημιουργία νέου ερωτηματολογίου μαζί με τον ταυτοποιημένο χρήστη (αν υπάρχει).
- **function edit(\$id)**
Επιστρέφει το view για την επεξεργασία ενός ερωτηματολογίου. Δέχεται σαν παράμετρο το id του ερωτηματολογίου. Αφού βρεθεί ο ταυτοποιημένος χρήστης και το συγκεκριμένο ερωτηματολόγιο, τσεκάρουμε τον ρόλο του χρήστη αν ο χρήστης είναι editor τσεκάρουμε αν το συγκεκριμένο ερωτηματολόγιο ανήκει στον χρήστη. Αν ναι τσεκάρουμε αν το ερωτηματολόγιο μπορεί να επεξεργαστεί, αν ναι τότε επιστρέφουμε το view edit-questionnaire μαζί με τον ταυτοποιημένο χρήστη και το ερωτηματολόγιο, αν όχι επιστρέφουμε μήνυμα στον χρήστη ότι δεν μπορεί να επεξεργαστεί ένα ερωτηματολόγιο το οποίο έχει ενεργοποιηθεί τουλάχιστον μία φορά. Αν ο χρήστης είναι admin ή Super admin δεν χρειάζεται να τσεκάρουμε αν του ανήκει το συγκεκριμένο ερωτηματολόγιο.
- **function postEdit(Request \$request)**
Γίνεται η επεξεργασία του ερωτηματολογίου. Δέχεται σαν παραμέτρους τα δεδομένα από τη φόρμα επεξεργασίας του ερωτηματολογίου. Τα δεδομένα από τη φόρμα περνάνε από validation όπου τα πεδία id title και question είναι υποχρεωτικά. Στη συνέχεια τσεκάρουμε τον ρόλο του χρήστη για να είμαστε σίγουροι ότι δεν θα επεξεργαστεί ερωτηματολόγιο που δεν έχει πρόσβαση (αν είναι editor) και τσεκάρουμε αν το ερωτηματολόγιο μπορεί να επεξεργαστεί. Αφού ολοκληρωθεί η διαδικασία ενημερώνουμε τον χρήστη για το αποτέλεσμα με σχετικό μήνυμα πίσω στο questionnaires view.
- **postNew(Request \$request)**
Η διαδικασία εισαγωγής νέου ερωτηματολογίου στη βάση δεδομένων. Δέχεται σαν παραμέτρους τα δεδομένα από τη φόρμα εισαγωγής νέου

ερωτηματολογίου. Τα δεδομένα περνάνε από validation υποχρεωτικά όλα τα πεδία (title και question). Με την ολοκλήρωση της διαδικασίας ο χρήστης επιστρέφει στο questionnaire view με σχετικό μήνυμα.

- **function postDisplay(Request \$request)**

Ορίζει αν το ερωτηματολόγιο θα είναι ορατό ή όχι από τους χρήστες του web app. Δέχεται σαν παράμετρο το id του ερωτηματολογίου και αλλάζει την τιμή του πεδίου display στον πίνακα questionnaires. Επιστρέφει το σχετικό μήνυμα στον χρήστη και μια Boolean μεταβλητή changeButtons που τη χρησιμοποιούμε για να αλλάξουμε το αντίστοιχο κουμπί στο view, καθώς η κλήση σε αυτό το function γίνεται με ajax request.

- **postActive(Request \$request)**

Αλλάζει την κατάσταση active του ερωτηματολογίου (αν είναι active είναι ορατό από τους χρήστες του mobile application και μπορούν να το συμπληρώσουν. Με την αλλαγή του ερωτηματολογίου σε active αλλάζει το πεδίο editable σε false και το ερωτηματολόγιο δεν μπορεί πια να επεξεργαστεί ενώ αν υπάρχει κάποιο άλλο ερωτηματολόγιο με active true, αυτό γίνεται false, καθώς μόνο ένα ερωτηματολόγιο μπορεί να είναι ενεργό. Με την ενεργοποίηση ενός ερωτηματολογίου πρέπει να ενημερωθούν οι χρήστες της mobile εφαρμογής ότι ένα νέο ερωτηματολόγιο είναι διαθέσιμο. Όλα τα firebase tokens από τη βάση στέλνονται στη function sendNotification() μαζί με το μήνυμα που θέλουμε να σταλεί στους χρήστες. Η function επιστρέφει ένα μήνυμα στον χρήστη ανάλογα με την κατάσταση του ερωτηματολογίου και μια Boolean μεταβλητή για την αλλαγή του αντίστοιχου κουμπιού στο view καθώς και εδώ η κλήση γίνεται με ajax request.

- **function sendNotification(\$devices, \$message)**

Στέλνει το notification στις ενεργοποιημένες συσκευές. Δέχεται παραμέτρους τα tokens των συσκευών και το μήνυμα. Η κλήση στο firebase cloud messaging γίνεται μέσω curl, και χρειάζονται τρεις μεταβλητές \$url, \$fields και \$headers. Το url είναι το url firebase cloud messaging (fcm) <https://fcm.googleapis.com/fcm/send>, η μεταβλητή fields είναι array με registration_ids τα firebase_tokens και ένα sub_array "notification" με title τον τίτλο του notification και body το μήνυμα. Η μεταβλητή headers, είναι και αυτή array και αφορά τα headers της κλήσης. Πρέπει να οριστεί το

“Authorization:key=” και το Content-Type το οποίο είναι “application/json” Το key βρίσκεται στο αρχείο .env της Laravel, καθώς δεν είναι καλή πρακτική κλειδιά τρίτων εφαρμογών να είναι hardcoded. Κινδυνεύουν να διαρρεύσουν σε οποιονδήποτε με τον διαμοιρασμό της εφαρμογής πχ στο github.

- **function getActiveQuestionnaire()**

Επιστέφει το ενεργό ερωτηματολόγιο (αν υπάρχει) στο mobile application.

- **function postDelete(Request \$request)**

Διαγράφει ένα ερωτηματολόγιο μαζί με τις απαντήσεις του από τη βάση δεδομένων. Δέχεται παράμετρο το id του ερωτηματολογίου, το οποίο γίνεται validate (υποχρεωτικό) και στη συνέχεια αφού βρεθεί το συγκεκριμένο ερωτηματολόγιο, τσεκάρουμε τον ρόλο του χρήστη αν ο χρήστης είναι Admin ή Super admin το ερωτηματολόγιο και οι απαντήσεις του διαγράφονται, ενώ αν είναι editor τσεκάρουμε να δούμε αν το ερωτηματολόγιο του ανήκει για να μπορέσει να το διαγράψει. Σε κάθε περίπτωση επιστρέφουμε το αντίστοιχο μήνυμα στον χρήστη. Η κλήση εδώ γίνεται με ajax request ο από το view.

- **postAnswer(Request \$request)**

Αποθηκεύει τις απαντήσεις του ερωτηματολογίου, οι οποίες έρχονται από το mobile application. Όλα τα πεδία είναι υποχρεωτικά και εφόσον ολοκληρωθεί επιτυχώς το validation με επιτυχία αποθηκεύονται οι απαντήσεις στον πίνακα answers, η όλη διαδικασία είναι κλεισμένη σε ένα try – catch block, καθώς θέλουμε να πιάσουμε το exception σε περίπτωση που ο χρήστης έχει ήδη απαντήσει στο συγκεκριμένο ερωτηματολόγιο. Έτσι λοιπόν θα πρέπει να βρούμε το error code του exception και να τσεκάρουμε αν είναι 1062 (duplicate entry) (Shreevatsa, 2017, stackOverflow).

Το function επιστρέφει σχετικό μήνυμα στον χρήστη καθώς και μια Boolean μεταβλητή “completed”.

- **UserController**

Περιέχει όλα τα functions για την προβολή, δημιουργία, επεξεργασία και διαγραφή των χρηστών του web application.

- **function index()**

Βρίσκει όλους τους users, και τους ρόλους εκτός από του Super admin. Επιστρέφει το users view μαζί με όλους τους users, τον ταυτοποιημένο χρήστη και τους ρόλους.

- **function getNew()**

Επιστρέφει τη view για τη δημιουργία του νέου χρήστη μαζί με τον ταυτοποιημένο χρήστη.

- **function register(Request \$request)**

Αποθηκεύει τον νέο χρήστη στη βάση δεδομένων. Δέχεται παραμέτρους από την φόρμα δημιουργίας νέου χρήστη. Και τα πεδία περνάνε από validation, όπου όλα τα πεδία είναι υποχρεωτικά, το role_id πρέπει να είναι από 1 έως 3 (editor – admin – super admin), το email πρέπει να είναι έγκυρης μορφής και μοναδικό για τους χρήστες και το password πρέπει να είναι ανάμεσα σε 6 και 12 χαρακτήρες. Ο κωδικός γίνεται hash πριν εισαχθεί στη βάση. Επιστρέφει το users view μέσω redirect, με σχετικό μήνυμα.

- **function updateRole(Request \$request)**

Αλλάζει τον ρόλο ενός χρήστη. Η κλήση γίνεται με ajax request από το view. Δέχεται παραμέτρους το id του χρήστη και τον ρόλο που πρέπει να εκχωρηθεί. Και τα δύο πεδία είναι απαραίτητα, ενώ ο ρόλος μπορεί να είναι μόνο 1 ή 2 (Δεν μπορεί να εκχωρηθεί ρόλος 2 (Super admin). Αν ο ταυτοποιημένος χρήστης είναι super admin μπορεί να εκχωρήσει οποιονδήποτε από τους δύο ρόλους σε κάποιον άλλον χρήστη, αν όχι μπορεί να μετατρέψει μόνο έναν editor σε admin και όχι το αντίστροφο. Η function επιστρέφει το ανάλογο μήνυμα στον χρήστη.

- **function getProfile()**

Επιστρέφει το profile view για την επεξεργασία του χρήστη.

- **function postProfile(Request \$request)**

Αλλάζει τα στοιχεία του χρήστη στη βάση δεδομένων. Δέχεται παραμέτρους τα πεδία από την αντίστοιχη φόρμα του profile view, και όλα τα στοιχεία είναι υποχρεωτικά το email πρέπει να είναι μοναδικό ως προς τους users μπορεί όμως να είναι το ίδιο με το υπάρχων του χρήστη. Η function επιστρέφει πίσω στο profile view τον χρήστη με σχετικό μήνυμα.

- **postPassword(Request \$request)**

Αλλάζει τον κωδικό του χρήστη. Δέχεται σαν παραμέτρους τα πεδία από την αντίστοιχη φόρμα στο profile view και όλα τα πεδία είναι υποχρεωτικά. Το

πεδίο `new_password` πρέπει να είναι ανάμεσα σε 6 και 12 χαρακτήρες, ενώ το πεδίο `confirm_password` πρέπει να είναι ακριβώς ίδιο με το πεδίο `new_password`. Ο νέος κωδικός γίνεται hash πριν εισαχθεί στη βάση δεδομένων. Ο χρήστης ανακατευθύνεται πίσω στο profile view με σχετικό μήνυμα.

- **function postDelete(Request \$request)**

Διαγράφει έναν χρήστη από τη βάση δεδομένων. Δέχεται σαν παράμετρο το `id` του χρήστη, το οποίο δεν μπορεί να είναι 1. Το ένα είναι πάντα ο Super admin ο οποίος δημιουργείται αυτόματα κατά το migration και δεν μπορεί να διαγραφεί. Μόνο ο super admin μπορεί να διαγράψει admins, ενώ οι admins μπορούν να διαγράψουν μόνο editors. Σε κάθε διαγραφή χρήστη τα ερωτηματολόγια που έχει αυτός δημιουργήσει περνάνε στην κυριότητα του Super admin. Η κλήση εδώ γίνεται με ajax request και η function επιστρέφει σχετικό μήνυμα μαζί με μία Boolean μεταβλητή “completed” για να γίνει update ο DOM του view.

2.6.3. Migrations

Στο folder migrations δημιουργούνται τα αρχεία με τα οποία η Laravel δημιουργεί τους πίνακες και τα πεδία σε αυτούς. Έχουμε για κάθε πίνακα που περιγράψαμε πιο πάνω το αντίστοιχο migration. Κάθε migration περιέχει δύο κύρια functions `up()` και `down()`. Στο function `up()` δημιουργείται ο πίνακας και τα πεδία ενώ το `down()` χρησιμεύει για να διαγραφεί ο πίνακας. Κάποια βασικά στοιχεία όπως ο χρήστης Super admin, τα φύλλα και οι ηλικιακές ομάδες εισάγονται κατά τη δημιουργία των πινάκων από το migration.

- **Create_users_table**

- `id`
- `first_name (string)`
- `last_name (string)`
- `email (string, unique)`
- `email_verified_at (timestamp, nullable)`
- `password (string)`
- `rememberToken`
- `timestamps (εισάγει τα πεδία created_at, updated_at)`

Στο migration αυτό εισάγουμε και τα στοιχεία του χρήστη Super admin.

- **Create_password_resets_table**
 - email (string, index)
 - token (string)
 - created_at (timestamps, nullable)

Δημιουργείται by default από τη Laravel

- **Create_questionnaires_table**
 - Id
 - User_id (bigInteger)
 - title (string)
 - questions (longText, nullable)
 - editable (Boolean, default = 1) true
 - display (Boolean default = 0) false
 - active (Boolean, default = 0) false
 - timestamps
- **Create_roles_table**
 - Id
 - name (string)
 - timestamps
- **create_firebase_tokens_table**
 - id
 - token (string)
 - timestamps
- **create_answers_table**
 - id
 - gender_id (bigInteger)
 - age_group_id (bigInteger)
 - questionnaire_id (bigInteger)
 - token_id (bigInteger)
 - answers (longText, nullable)
 - timestamps
 - questionnaire_id, token_id (unique) Primary key
- **create_genders_table**
 - id
 - name (string)

- timestamps

Στο migration αυτό εισάγουμε τα τρία φύλα (male, female, other)

- **create_age_groups_table**

- id
- group (string)
- timestamps

Στο migration αυτό εισάγουμε τις ηλικιακές ομάδες (< 18, 18 - 30, 31 – 45, 46 – 55, > 55)

2.6.4. Routes

Για να γίνει η πραγματική σύνδεση των URLS με τους controllers και τα functions αυτών η Laravel χρησιμοποιεί τα route files. Υπάρχουν δύο ειδών routes το web route (web.php), και το api route (api.php). Η διαφορά μεταξύ τους είναι ότι στο web route ορίζονται τα routes που έρχονται από την ίδια την εφαρμογή (Laravel) και κάθε φόρμα πρέπει να περιέχει το CSRF token, ενώ το api route είναι φτιαγμένο να δέχεται κλήσης από οπουδήποτε στη συγκεκριμένη περίπτωση από το mobile application. (Laravel docs 8.x, 2021)

Το CSRF token (Cross Site Request Forgery) υπάρχει για να εμποδίζει requests requests από άλλα sites να αλλάξουν τη βάση δεδομένων πχ μέσω της αντιγραφής της φόρμας μας. Το token δημιουργείται για κάθε χρήστη κάθε φορά που χρησιμοποιεί την εφαρμογή και αποθηκεύεται στο user session. Σε κάθε υποβολή φόρμας η Laravel τσεκάρει αν το token που έχει υποβληθεί είναι ίδιο με το token του χρήστη στο session. (Laravel docs 8.x, 2021)

Για τη δημιουργία των route χρειάζονται κάποιοι παράμετροι

1. Η μέθοδος κλήσης (GET, POST, DELETE, PUT PATCH, OPTIONS)
2. Το URL
3. Ένα array που περιλαμβάνει την κλάση του controller και τη μέθοδο που θέλουμε να χρησιμοποιήσουμε.

Πχ για να κατευθύνουμε τον χρήστη στην αρχική σελίδα

`Route::get('/', [HomeController::class, 'index']);`

Όπου η function index του HomeController επιστρέφει το view της αρχικής σελίδας.

2.6.5. Middlewares

Για την προστασία των routes από μη εξουσιοδοτημένη πρόσβαση η Laravel χρησιμοποιεί τα middlewares. Για παράδειγμα το middleware τσεκάρει αν ένας χρήστης είναι ταυτοποιημένος πριν προχωρήσει με κάποιο request και αν δεν είναι τον ανακατευθύνει πίσω στη σελίδα login. Όλα τα απαραίτητα middleware είναι δημιουργημένα by default, στην περίπτωση μας όμως πρέπει να δημιουργηθεί ένα νέο middleware που να τσεκάρει αν ο χρήστης ανήκει στους Admins ή στον Super admin. (Laravel docs 8.x, 2021)

Δημιουργήθηκε λοιπόν το middleware **AccessAdmin** στο οποίο τσεκάρουμε αν ο χρήστης είναι ανήκει στους Admins ή στον Super admin. Αν ναι, προχωράει με το request, αν όχι τον ανακατευθύνει στην αρχική σελίδα.

Για να γίνει το check δημιουργούμε ένα νέο Gate function boot() του app/Providers/AuthServiceProvider με το όνομα is-admin. Το gate καλεί την function hasAdminRole() από το User model η οποία επιστρέφει true αν ο χρήστης είναι Admin ή Super admin και false αν είναι editor. (Laravel docs 8.x, 2021). Στο middleware τσεκάρουμε το αποτέλεσμα του Gate.

Για να μπορούμε να χρησιμοποιήσουμε το middleware πρέπει να το δηλώσουμε στο kernel.php στο array \$routeMiddleware με κάποιο όνομα, στην περίπτωση μας auth.admin. Στη συνέχεια για να προστατέψουμε τα routes μας με το συγκεκριμένο middleware τα κάνουμε group υπό το middleware. Τα routes που θέλουμε να είναι public τα αφήνουμε εκτός middleware, τα routes που δεν χρειάζεται ο χρήστης να είναι Admin ή Super admin τα κάνουμε group υπό το default auth middleware, ενώ τα routes που θέλουμε να έχουν πρόσβαση μόνο οι Admin ή ο Super Admin τα κάνουμε group με το νέο auth-admin middleware.

2.6.6. Views

Τα view είναι αυτό που βλέπει ο χρήστης, η Laravel χρησιμοποιεί την επέκταση blade για να καθορίσει ένα αρχείο ως view και να μπορεί να χρησιμοποιηθεί ένα ξεχωριστό syntax για την εκτύπωση μεταβλητών if και for statements της php. Το folder structure των views έχει ως εξής:

- Login.blade.php (Σελίδα login)
- **Incl** (folder)

Περιέχει το header και το footer της εφαρμογής τα οποία γίνονται include σε κάθε blade αρχείο.

- **Public** (folder)

Περιέχει τα views που είναι προσβάσιμα από όλους τους χρήστες της εφαρμογής χωρίς να κάνουν login.

- [home.blade.php](#) (Αρχική σελίδα)
- [questionnaire.blade.php](#) (σελίδα με τα διαγράμματα των απαντήσεων ενός ερωτηματολογίου)

- **users** (folder)

Περιέχει τα views όπου ο χρήστης πρέπει να κάνει login για να έχει πρόσβαση

- [edit-questionnaire.blade.php](#) (Σελίδα επεξεργασίας ερωτηματολογίου)
- [new-questionnaire.blade.php](#) (Σελίδα δημιουργίας νέου ερωτηματολογίου)
- [new-user.blade.php](#) (Σελίδα δημιουργίας νέου χρήστη)
- [profile.blade.php](#) (Σελίδα επεξεργασίας στοιχείων του χρήστη)
- [questionnaire.blade.php](#) (Σελίδα όπου εμφανίζονται τα ερωτηματολόγια)
- [users.blade.php](#) (Σελίδα που εμφανίζονται όλοι οι χρήστες)

Για το layout των views έχει χρησιμοποιηθεί το bootstrap 4 καθώς και κάποια custom classes CSS στον φάκελο /public/assets/css/fivos.css. Στον ίδιο φάκελο έχουν συμπεριληφθεί και τα αρχεία CSS του bootstrap. Ενώ τα αρχεία JavaScript βρίσκονται στον φάκελο /public/assets/js/, ενώ όλα τα εικονίδια που χρησιμοποιήθηκαν στην εφαρμογή βρίσκονται στον φάκελο /public/assets/img/.

2.7. Περιγραφή λειτουργίας mobile application.

Η χρήση του mobile application είναι πάρα πολύ απλή. Η εφαρμογή αποτελείται από τρεις οθόνες, τις οποίες ο χρήστης μπορεί να αλλάζει μέσω ενός tab layout στο κάτω μέρος της οθόνης. Στην πρώτη οθόνη ο χρήστης βλέπει τα στοιχεία από την αρχή της πανδημίας για την Ελλάδα. Τα στοιχεία είναι χωρισμένα ανά ημερομηνία από την αρχή της πανδημίας και παρουσιάζονται με cardViews. Για κάθε ημερομηνία ο χρήστης βλέπει το σύνολο των κρουσμάτων και των θανάτων μέχρι τη συγκεκριμένη ημερομηνία καθώς και την καθημερινή αύξηση αυτών. Υπάρχει δυνατότητα αναζήτησης στο πάνω μέρος της οθόνης, όπου ο χρήστης μπορεί να επιλέξει την ημερομηνία που θέλει και να δει τα αντίστοιχα στοιχεία.

Πανδημία		2021-09-04	
		Κρούσματα	Θάνατοι
		598667	13843
		↑ 2284	↑ 106
		2021-09-03	
		Κρούσματα	Θάνατοι
		596383	13737
		↑ 2715	↑ 35
		2021-09-02	
		Κρούσματα	Θάνατοι
		593668	13702
		↑ 2836	↑ 33

Πανδημία

Quiz

Πληροφορίες

Εικόνα 15 Οθόνη εφαρμογής με τα στοιχεία της πανδημίας.

Η δεύτερη οθόνη (Quiz) εμφανίζει στον χρήστη το διαθέσιμο ερωτηματολόγιο. Την πρώτη φορά που ο χρήστης θα ανοίξει την συγκεκριμένη οθόνη θα πρέπει να επιλέξει το φύλο του και την ηλικιακή ομάδα που ανήκει. Αφού δώσει αυτές τις υποχρεωτικές, μπορεί να δει και να συμπληρώσει το ερωτηματολόγιο (αν αυτό είναι διαθέσιμο). Κάθε χρήστης μπορεί να συμπληρώσει ένα ερωτηματολόγιο μόνο μία φορά. Οι απαντήσεις είναι της μορφής ΣΩΣΤΟ – ΛΑΘΟΣ, και είναι όλες υποχρεωτικές, δεν μπορεί να γίνει υποβολή αν δεν έχουν απαντηθεί όλες οι ερωτήσεις. Ο χρήστης μπορεί να μεταβεί στην επόμενη ή την προηγούμενη ερώτηση μέσω αντίστοιχων κουμπιών κάτω από τις ερωτήσεις ενώ υπάρχει και η δυνατότητα πλοήγησης σε όποια ερώτηση θέλει από τα πάνω τετράγωνα κουμπιά. Η αλλαγή χρώματος στα πάνω κουμπιά δείχνει ποιες ερωτήσεις έχουν απαντηθεί και ποιες όχι. Ενώ στο κάτω μέρος της οθόνης υπάρχει το κουμπί “Υποβολή” όπου αν έχουν απαντηθεί όλες οι ερωτήσεις υποβάλλονται στο web app, το οποίο τσεκάρει μέσω του firebase key που έχει δημιουργηθεί για τη συγκεκριμένη συσκευή αν ο συγκεκριμένος χρήστης έχει απαντήσει ξανά το συγκεκριμένο ερωτηματολόγιο. Αν όχι το web app επιστρέφει μήνυμα ότι οι απαντήσεις υποβλήθηκαν με επιτυχία, ενώ αν ο χρήστης έχει απαντήσει ξανά ενημερώνεται με σχετικό μήνυμα.

Quiz

Πριν απαντήσετε στο ερωτηματολόγιο θα χρειαστούμε μερικές πληροφορίες.

Ποιο είναι το φύλο σας;

☐ Άνδρας

☐ Γυναίκα

☐ Άλλο

Σε ποια ηλικιακή ομάδα ανήκετε;

☐ < 18

☐ 18 – 30

☐ 31 – 45


Πανδημία


Quiz


Πληροφορίες

Εικόνα 16: Προσωπικές πληροφορίες χρήστη.

Quiz

1

2

Οι μη-συμπτωματικοί ασθενείς με COVID-19 δεν είναι μεταδοτικοί.


Πανδημία

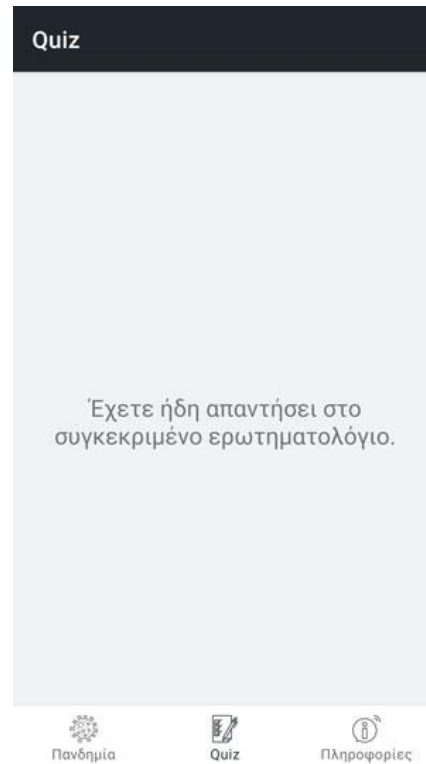

Quiz


Πληροφορίες

Εικόνα 17: Ερωτηματολόγιο.



Εικόνα 18: Επιτυχής υποβολή απαντήσεων.



Εικόνα 19: Ο χρήστης έχει ήδη απαντήσει στο ερωτηματολόγιο.

Η Τρίτη οθόνη της εφαρμογής παρέχει πληροφορίες σχετικά με την εφαρμογή καθώς και ένα email για επικοινωνία των χρηστών με τον προγραμματιστή σχετικά με οποιαδήποτε απορία ή πρόβλημα.



Εικόνα 20: Πληροφορίες εφαρμογής.

2.8. Άδειες εφαρμογής

Η μοναδική άδεια που χρειάζεται η εφαρμογή για να λειτουργεί σωστά, είναι η άδεια πρόσβασης στο internet, η οποία έχει δηλωθεί στο manifest file

```
<uses-permission  
android:name="android.permission.INTERNET" />
```

Μαζί με τη άδεια πρόσβασης στο internet χρησιμοποιούμε και την άδεια εναλλαγής κατάστασης δικτύου. Χρειαζόμαστε αυτή την άδεια για να γνωρίζουμε την κατάσταση του δικτύου πριν προχωρήσουμε στην κλήση των APIs.

```
<uses-permission  
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

2.9. 3rd party libraries

Η εφαρμογή χρησιμοποιεί δύο εξωτερικά library, το ένα για την πραγματοποίηση κλήσεων στα APIs και το άλλο για την ενσωμάτωση σε αυτή της υπηρεσίας firebase messaging.

Για την πραγματοποίηση κλήσεων στα APIs χρησιμοποιούμε το Retrofit. Το Retrofit είναι μία βιβλιοθήκη για Android σε Java και Kotlin η οποία έχει αναπτυχθεί από τη Square και κάνει πιο εύκολες τις κλήσεις σε APIs. Τα Requests στέλνονται με OkHttp. (codepath, 2021).

Για την αποστολή notifications στους χρήστες της εφαρμογής χρησιμοποιούμε το Firebase Cloud Messaging. Είναι μια δωρεάν υπηρεσία που παρέχετε από την google με εξαιρετικό documentation και παραδείγματα. Για τη χρήση της υπηρεσίας απαιτείται λογαριασμός στο firebase console <https://firebase.google.com/>.

2.10. Εκδόσεις android που υποστηρίζεται η εφαρμογή.

Η εφαρμογή τρέχει σε android 6.0 και μετά (API level 23), ενώ παρατηρήθηκε ότι στο android 11 (API level 30), αποτυγχάνουν τα requests του retrofit. Αφού το retrofit χρησιμοποιεί το OkHttp και λογικά το API 30 του android απαιτεί συγκεκριμένη έκδοση θα πρέπει να εισαχθεί η τελευταία έκδοση της OkHttp3 library στο gradle file της εφαρμογής.

```
implementation 'com.squareup.okhttp3:okhttp:4.9.0'
```

2.11. File and code structure εφαρμογής

Ο κώδικας της εφαρμογής χωρίζεται σε αρκετά κομμάτια για την καλύτερη κατανόηση και συντήρησή του. Παρακάτω ακολουθεί το folder structure της εφαρμογής.

- **Api**

Περιέχει τα Retrofit instances και τα calls στα δύο API με τις μεθόδους αυτόν (POST, GET).

- **AppiCalls** (interface)

Περιέχει τις μεθόδους και τα URL για την κλήση των APIs

- **CasesRetrofitInstance** (Object)

Το retrofit instance για την κλήση του covid-19-greece api.

- **MyApiRetrofitInstance** (Object)

Το retrofit instance για την κλήση του API που δημιουργήθηκε για τους σκοπούς αυτής της εφαρμογής.

- **Libraries**

Περιέχει βοηθητικές κλάσεις για την αποθήκευση δεδομένων και επεξεργασία αυτών κατά την εκτέλεση της εφαρμογής.

- **CasesLibrary** (Class)

Περιέχει δύο ArrayLists (casesList, searchList). Το casesList κρατάει τα όλα τα δεδομένα από την κλήση του covid-19-greece api, ενώ το searchList τα δεδομένα από την αναζήτηση που κάνει ο χρήστης.

- QuizLibrary (class)

Περιέχει το arrayList fetchedQuestions στο οποίο αποθηκεύονται οι ερωτήσεις του ερωτηματολογίου, ένα array όπου αποθηκεύονται οι απαντήσεις και μι function getQuestion() που επιστρέφει την ερώτηση ανάλογα με το position.

- **Model**

Περιέχει τις κλάσεις που καθορίζουν τα πεδία των κλήσεων στα APIs.

- **Answer** (data class)

Περιέχει τα πεδία που πρέπει να σταλούν στο API για την αποστολή των απαντήσεων σε ένα ερωτηματολόγιο.

- questionnaire_id: int
 - gender_id: int
 - age_group_id: int

- token: String
 - answers: Array<String>
- **Case** (data class)

Περιέχει τα πεδία που επιστέφονται από το covid-19-greece api για κάθε ημερομηνία.

 - confirmed: Int
 - date: String
 - deaths: Int
- **Cases** (data class)

Περιέχει τη λίστα με όλες της ημερομηνίες από το covid-19-greece api

 - cases: List<Case>
- **Completed** (data class)

Περιέχει την επιστροφή των POST requests από το δικό μας API.

 - completed: Boolean (Επιτυχία ή όχι εισαγωγής)
 - msg: String (Το μήνυμα για τον χρήστη)
- **Questionnaire** (data class)

Περιέχει τα πεδία που επιστέφονται όταν η εφαρμογή ζητάει το διαθέσιμο ερωτηματολόγιο.

 - id: Int
 - title: String
 - questions: ArrayList<String>
- **Token** (data class)

Περιέχει το πεδίο token (firebase token) που θα εισαχθεί στη βάση δεδομένων του API.

 - token: String
- **operations**

Περιέχει κλάσεις με διαδικασίες που δεν θέλουμε ή δεν μπορούν να έχουμε στο βασικό Activity ή στα επιμέρους Fragments.

 - **Calls** (class)

Περιέχει τα functions που καλούν τα APIs και δεν μπορούμε να τα έχουμε στο αντίστοιχο Fragment.

 - postToken (function)

Δέχεται το context και το token και στη συνέχεια καλεί το function pushToken από το Apicalls interface μέσω του MyApiRetrofitInstance.

Αν το post request εκτελεστεί με επιτυχία εμφανίζει μήνυμα στον χρήστη “App registered” αν όχι εμφανίζει “App not registered”.

- **Message (class)**

Εμφανίζει Toast messages στον χρήστη.

- **Message (function)**

Δέχεται το context και ένα μήνυμα. Εμφανίζει το μήνυμα.

- **RecyclerViewAdapters**

Περιέχει τον adapter για το recyclerView που εμφανίζει της πληροφορίες σχετικά με την πανδημία στην αρχική οθόνη της εφαρμογής.

- **CasesAdapter (class)**

Δέχεται το context και τις δύο λίστες (searchList, casesList) Αρχικά δημιουργούνται δύο int μεταβλητές όπου θα κρατάνε το αποτέλεσμα για την καθημερινή αύξηση κρουσμάτων και θανάτων, καθώς το covid-19-greece api επιστρέφει για κάθε μέρα το σύνολο από την αρχή της πανδημίας και για τις δύο περιπτώσεις.

- **ViewHolder (sub class)**

Δηλώνονται τα widgets που θα τοποθετηθούν τα δεδομένα.

- **onCreateViewHolder (override function)**

Γίνετε inflate το layout το οποίο περιέχει τα widgets του ViewHolder class.

- **onBindViewHolder (override function)**

δέχεται για κάθε ημερομηνία (από τη λίστα searchList) τον ViewHolder, καθώς και τη θέση στην οποία βρίσκεται η λίστα. Σε αυτό το function τοποθετούνται τα δεδομένα για να μπορέσει να τα δει ο χρήστης. Ο υπολογισμός των καθημερινών κρουσμάτων και θανάτων γίνεται με αφαίρεση αυτών από την προηγούμενη ημερομηνία.

- **getItemCount (override function)**

Επιστρέφει το μέγεθος της λίστας με τα δεδομένα.

- **Repositories**

- **Repository (class)**

Επιστρέφει για κάθε function (getCases(), getQuiz(), pushAnswers()), το αντίστοιχο retrofit instance με το function που πρέπει να καλέσει από τα apiCalls.

- **Services**

- **CycleObserver** (class extends LifecycleObserver)

Δέχεται context. Μας δίνει τη δυνατότητα να γνωρίζουμε αν το activity έχει δημιουργηθεί. Όταν το activity ξεκινήσει τσεκάρουμε αν υπάρχει δίκτυο και αν όχι εμφανίζουμε σχετικό μήνυμα.
- **NotificationService** (class)

Διαχειρίζεται τα notifications που λαμβάνει η εφαρμογή

 - **onMessageReceived** (override function)

δέχεται το notificationMessage και καλεί τη function sendNotification η οποία διαχειρίζεται πως θα εμφανιστεί το notification.
 - **onNewToken** (override function)

Δέχεται το token που δημιουργείται για την συγκεκριμένη συσκευή από την υπηρεσία firebase. Κρατάμε αυτό το token στα shared preferences της εφαρμογής και στη συνέχεια καλούμε τη function postToken από την class calls στον φάκελο operations για να εισάγουμε το token στη βάση δεδομένων του API μας.
 - **sendNotification** (function)

Διαχειρίζεται τον τρόπο με τον οποίο θα εμφανιστεί το notification στη συσκευή του χρήστη. Καλείται από τη function onMessageReceived().
- **UiOperations** (class)

Διαχειρίζεται τα updates του user interface. Ο λόγος που βρίσκονται σε ξεχωριστό class είναι γιατί διαφορετικά θα μεγάλωναν πολύ τα fragments. Δέχεται μόνο το context σαν παράμετρο.

 - **Toggle** (function)

Δέχεται μία Boolean και αναλόγως εμφανίζει ή εξαφανίζει το loading screen της εφαρμογής.
 - **Message** (function)

Δέχεται ένα string (μήνυμα) και εμφανίζει μόνο το textView από το loading screen για να παρουσιαστεί το αποτέλεσμα των κλήσεων (APIs) ή μήνυμα σφάλματος στον χρήστη.
 - **answerButtons** (function)

Δέχεται τα δύο κουμπιά (TRUE – FALSE) για κάθε ερώτηση καθώς και μία Boolean selection. Αλλάζει το χρώμα των κουμπιών ανάλογα με το selection (επιλογή χρήστη).
 - **nextPrevVisibility** (function)

Δέχεται τα δύο κουμπιά (prev – next), τη θέση στην οποία βρίσκεται το ερωτηματολόγιο καθώς και το μέγεθος αυτού. Αν το ερωτηματολόγιο είναι στην πρώτη θέση (Πρώτη ερώτηση) κρύβει το κουμπί που γυρνάει σε προηγούμενη ερώτηση, το εμφανίζει όμως από τη δεύτερη θέση και μετά. Παρομοίως για το κουμπί της επόμενης ερώτησης, εξαφανίζεται μόνο όταν το ερωτηματολόγιο βρίσκεται στην τελευταία ερώτηση.

- **showUserInfo (function)**

Δέχεται ένα Boolean show, και αν είναι true εμφανίζει την οθόνη όπου ο χρήστης πρέπει να συμπληρώσει τις προσωπικές πληροφορίες (φύλο και ηλικιακή ομάδα).

- **changeTopNavigationButtonsColor (function)**

Δέχεται λίστα με κουμπιά, array με τις απαντήσεις και το και το position στο οποίο βρίσκεται η λίστα (κουμπί). Αλλάζει το χρώμα των κουμπιών (αν έχει δοθεί απάντηση) καθώς και το χρώμα στο κουμπί της τρέχουσας ερώτησης.

- **ui**

Περιέχει τα class του activity και των fragment

- **activity**

περιέχει το class της MainActivity της εφαρμογής.

- **MainActivity (class)**

Η οθόνη με τις προσωπικές πληροφορίες του χρήστη ανήκει στο MainActivity. Οι επιλογές του χρήστη αποθηκεύονται στις public μεταβλητές gender και age για να αποθηκευτούν αργότερα στα shared preferences της εφαρμογής.

- **getScreenWidth (function)**

Επιστρέφει το width της οθόνης της συσκευής.

- **Dashboard**

Περιέχει τα class που διαχειρίζονται τις διαδικασίες που έχουν να κάνουν μες το ερωτηματολόγιο.

- **QuizFragment (class extends Fragment, LifecycleObserver)**

Το fragment όπου εμφανίζονται τα ερωτηματολόγια.

- **onConfigurationChanged (override function)**

Σε κάθε αλλαγή orientation καλεί την `getScreenWidth` του `MainActivity` και επανατοποθετεί τα κουμπιά πλοήγησης του ερωτηματολογίου.

- `createNavigationButtons` (function)
Δημιουργεί τα κουμπιά πλοήγησης τοποθετώντας τα σε ένα `GridLayout` όπου τα `rows` και τα `cols` καθορίζονται από το `width` της οθόνης και το πλήθος των κουμπιών (πλήθος ερωτήσεων).
- `prevQuestion` (function)
Μετακινεί τον χρήστη στην προηγούμενη ερώτηση.
- `nextQuestion` (function)
Μετακινεί τον χρήστη στην επόμενη ερώτηση
- `getAnswer` (function)

Τσεκάρει την απάντηση στην συγκεκριμένη ερώτηση και αναλόγως καλεί την `answerButton` του `UiOperations` για να αλλάξει το χρώμα στο κουμπί της απάντησης.

- `checkUserInfo` (function)
Τσεκάρει αν υπάρχει το `key gender` στα `shared preferences` και αναλόγως καλεί την `showUserInfo` του `UiOperations` για να εμφανίσει ή όχι την οθόνη με τις προσωπικές πληροφορίες του χρήστη.
- `getQuestions` (function)
Καλεί μέσω τη function `getQuiz` του `viewModel`, εμφανίζει την `loading screen` μέχρι να επιστρέψει δεδομένα. Αν βρεθεί ερωτηματολόγιο εμφανίζεται στον χρήστη αν όχι εμφανίζεται σχετικό μήνυμα. Θα εμφανιστεί μήνυμα και σε περίπτωση σφάλματος είτε από τη μεριά της συσκευής (Δεν υπάρχει δίκτυο) είτε από τη μεριά του API (λάθος URL, server error)
- `checkAnswers` (function)
Τσεκάρει αν έχουν απαντηθεί όλες οι ερωτήσεις.
- `postAnswers` (function)
Στέλνει της απαντήσεις καλώντας την `push answer` του `viewModel`. Μέχρι να ολοκληρωθεί η διαδικασία εμφανίζεται η

loading screen και ο χρήστης ενημερώνεται για το αποτέλεσμα με σχετικό μήνυμα (είτε αυτό είναι σφάλμα)

- **QuizViewModel** (class extends ViewModel())

Δέχεται το αντίστοιχο repository και περιέχει τα functions getQuiz() και pushAnswers. Τα οποία τρέχουν σε coroutine και ξεκινάνε τη διαδικασία της κλήσης των API και επιστρέφουν τα δεδομένα πίσω στο Fragment.

- **QuizViewModelFactory** (class)

Δέχεται το repository και επιστρέφει instance του viewModel

- **Cases**

Περιέχει τα class που είναι υπεύθυνα για την προβολή των πληροφοριών για την πανδημία.

- **CasesFragment** (class extends Fragment)

- onCreateOptionsMenu (override function)

Δέχεται το menu και τον Menu Inflater, δημιουργεί το μενού, το οποίο είναι ένα searchView και κάνει implement τα functions onQueryTextSubmit() η οποία επιστρέφει απλά false (δεν κάνει τίποτα) και onQueryTextChange(), όπου δέχεται το input του χρήστη και τσεκάρει αν υπάρχει στη στην casesList αν υπάρχουν δεδομένα τα εισάγει στη searchlist και ανανεώνει τον adapter.

- getCases (function)

Καλεί την getCases από το viewModel και επιστρέφει την απάντηση στον χρήστη. Όσο ο χρήστης περιμένει την απάντηση εμφανίζεται η loading screen. Αν η κλήση στο API είναι επιτυχής τότε φορτώνουν τα δεδομένα, αλλιώς εμφανίζεται σχετικό μήνυμα.

- **CasesViewModel** (class extends viewModel)

Δέχεται το αντίστοιχο repository και περιέχει το function getCases() το οποίο τρέχει σε coroutine και ξεκινάει τη διαδικασία κλήσης του API. Επιστρέφει το αποτέλεσμα.

- **CasesViewModelFactory** (class)

Δέχεται το repository και επιστρέφει instance του viewModel

- **Notifications**

Περιέχει το fragment με τις πληροφορίες για την εφαρμογή

- **Utils**

Περιέχει βοηθητικά classes για την διαχείριση των κλήσεων με το retrofit

- **Resource** (class)

Περιέχει τα functions success, error, και loading τα οποία χρησιμοποιούμε για να κάνουμε update το UI ανάλογα με το αποτέλεσμα, στα resource αυτά γίνεται assign μια enum value.

- **Status** (enum class)

Περιέχει τα values που κάνουμε assign στο από πάνω resource

- **NetworkLiveData** (class)

Το class αυτό περιέχει το function isOnline που επιστρέφει τιμή true αν είναι συνδεδεμένη η συσκευή με δεδομένα, wifi ή ethernet και το function getDetails() η οποία κάνει την ίδια δουλειά χωρίς να ξέρουμε όμως τον τύπο της σύνδεσης στη συσκευή.

2.12. Other resources

Τα επιμέρους layout files που δημιουργήθηκαν βρίσκονται στο φάκελο res -> layout και res -> menu για το searchBar του CasesFragment. Τα εικονίδια που χρησιμοποιήθηκαν βρίσκονται στον φάκελο res -> drawable, ενώ όλα τα μνήματα και κείμενα της εφαρμογής είναι στο αρχείο res -> values -> strings.xml. Επεξεργάστηκαν τα αρχεία res -> values -> themes -> themes.xml με τα απαραίτητα χρώματα της εφαρμογής. Δεν δίνεται προς το παρών η δυνατότητα χρήσης dark mode.

3. Προτάσεις βελτίωσης

Το Greek covid-19 API παρέχει πολύ περισσότερα δεδομένα από αυτά που χρησιμοποιεί η εφαρμογή αυτή τη στιγμή, όπως δεδομένα για την πορεία των εμβολιασμών ανά περιοχή και ημερομηνία. Θα μπορούσε ο χρήστης να επιλέξει τι θα δει μέσω ενός μενού.

Όσο αφορά τα ερωτηματολόγια αυτή τη στιγμή υπάρχει η δυνατότητα δημιουργίας ερωτήσεων του τύπου Σωστό – Λάθος. Με λίγη δουλειά και στις δύο εφαρμογές (web app και mobile app) θα μπορούσαμε να έχουμε ερωτήσεις με προκαθορισμένες απαντήσεις (όχι μόνο σωστό λάθος). Η ερευνητική ομάδα θα εισάγει τις πιθανές απαντήσεις για κάθε ερώτηση, και πίσω στην εφαρμογή τα κουμπιά των απαντήσεων θα δημιουργούνται δυναμικά.

Αυτή τη στιγμή μαζί με τις απαντήσεις ενός χρήστη αποθηκεύουμε το φύλο και την ηλικιακή ομάδα στην οποία ανήκει, εδώ θα μπορούσα να προστεθούν και οι γεωγραφικές

συντεταγμένες του χρήστη (εφόσον αυτός το επιθυμεί). Τα αποτελέσματα θα μπορούν να παρουσιάζονται σε ένα χάρτη και να ομαδοποιούνται ανάλογα με την περιοχή, δίνοντάς μας πολύ περισσότερο περιθώριο ανάλυσης και κατανόησης αυτών.

Τέλος για να εξασφαλίσουμε ότι οι χρήστες δεν θα απαντάνε πολλαπλές φορές σε ένα ερωτηματολόγιο χρησιμοποιούμε το firebase token που δημιουργείται από το firebase και είναι μοναδικό για κάθε συσκευή. Ενώ η λογική αυτή δείχνει να δουλεύει, ο χρήστης μπορεί να σβήσει τα δεδομένα της εφαρμογής η ακόμα να την απεγκαταστήσει και να την εγκαταστήσει ξανά. Σε αυτή την περίπτωση η συσκευή παίρνει νέο token και ο χρήστης μπορεί να απαντήσει ξανά στο ίδιο ερωτηματολόγιο.

Καλό θα ήταν στην εφαρμογή να υπάρχει και η απεικόνιση των αποτελεσμάτων των ερευνών με τη μορφή διαγραμμάτων όπως συμβαίνει στο web app.

Όσο αφορά το web app θα πρέπει να συνδεθεί με μια υπηρεσία mail ώστε να μπορεί να ταυτοποιηθεί το mail του χρήστη από το σύστημα. Η ταυτοποίηση των χρηστών μπορεί να γίνει με json web tokens (jwt). Υπάρχουν αρκετά πακέτα για τη Laravel (πχ sanctum).

4. Συμπεράσματα

Από την αρχή της πανδημίας η τεχνολογία έπαιξε σημαντικό ρόλο στην αντιμετώπισή της δημιουργήθηκαν πολλές εφαρμογές των οποίων η επιτυχία δεν έχει αποδειχθεί ακόμα και σήμερα. Στον σύγχρονο κόσμο η προστασία των προσωπικών δεδομένων στο διαδίκτυο γίνεται ολοένα και πιο δύσκολη με αποτέλεσμα την δυσκολία υιοθέτησης των contact tracing apps από τους χρήστες (πολίτες). Με την ενημέρωση του κόσμου πάνω στα θέματα της πανδημίας και της προστασίας από αυτή να αποτελεί σημαντικό πρόβλημα οι κυβερνήσεις θα πρέπει να επενδύσουν κυρίως σε ενημερωτικές εφαρμογές όπου θα μπορούν να εξάγουν συμπεράσματα για τις πεποιθήσεις των πολιτών σχετικά με την πανδημία και την διαχείριση αυτής. Τα συμπεράσματα θα πρέπει να αξιολογούνται ώστε να επανασχεδιάζεται η στρατηγική ενημέρωσης. Η επιβολή τη χρήσης των contact tracing apps ίσως φέρει αντίθετα αποτελέσματα και στην πορεία της πανδημίας, αλλά και στην ενημέρωση του κόσμου σχετικά με αυτή.

Η εφαρμογή που δημιουργήθηκε στα πλαίσια αυτής της εργασίας σέβεται την ιδιωτικότητα του χρήστη, και ο ίδιος παραμένει ανώνυμος. Αν χρειαστούν επιπλέον δεδομένα όπως η γεωγραφική του θέση αυτός θα πρέπει να το εγκρίνει. Οι απαντήσεις στις διάφορες έρευνες μέσω της εφαρμογής θα βοηθήσουν στη διεξαγωγή συμπερασμάτων τόσο στο σύνολο της επικράτειας, όσο και σε διαφορετικές περιοχές (σε περίπτωση που κρατάμε και την

γεωγραφική θέση του χρήστη). Τα δεδομένα αυτά μπορούν να χρησιμοποιηθούν για την επαναξιολόγηση της στρατηγικής αντιμετώπισης της πανδημίας.

Βιβλιογραφία

Άρθρα

Chan, E. Y., & Saqib, N. U. (2021). Privacy concerns can explain unwillingness to download and use contact tracing apps when COVID-19 concerns are high. *Computers in Human Behavior*, 119, 106718. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7840411/>

Cioffi, A., Lugi, C., & Cecannecchia, C. (2020). Apps for COVID-19 contact-tracing: Too many questions and few answers. *Ethics, Medicine and Public Health*, 15, 100575. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7421297/>

John Leon Singh, H., Couch, D., & Yap, K. (2020). Mobile Health Apps That Help With COVID-19 Management: Scoping Review. *JMIR Nursing*, 3(1), e20596. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7467120/>

Ιστός

FontAwesome. (n.d.). FontAwesome. Retrieved August 7, 2021, from <https://fontawesome.com/>

Foxandroid. (2021, May 11). *Filter RecyclerView using Search View - Kotlin || Implement Search view in RecyclerView using Kotlin*. YouTube. <https://www.youtube.com/watch?v=K5YnTvsVPRk>

Free Vector Icons and Stickers - Thousands of resources to download. (n.d.). Flaticon. Retrieved June 5, 2021, from <https://www.flaticon.com/>

GitHub - Covid-19-Response-Greece/covid19-greece-api:  A simple and fast API for tracking the coronavirus (COVID-19) outbreak in Greece GR. (n.d.). GitHub. Retrieved July 8, 2021, from <https://github.com/Covid-19-Response-Greece/covid19-greece-api>

<https://covid19.apple.com/contacttracing>. (n.d.). Centers for Disease Control and Prevention. Retrieved September 6, 2021, from <https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/coronavirus-self-checker.html>

Installation - Laravel - The PHP Framework For Web Artisans. (n.d.). Laravel. Retrieved June 6, 2021, from <https://laravel.com/docs/8.x/>

Kotlin and Android |. (n.d.). Android Developers. Retrieved August 6, 2021, from <https://developer.android.com/kotlin>

Lukasgeiter, Faruk. (2010). *Laravel catch Eloquent "Unique" field error*. Stack Overflow. <https://stackoverflow.com/questions/27878719/laravel-catch-eloquent-unique-field-error>

Mobile contact tracing apps in EU Member States. (n.d.). European Commission - European Commission. Retrieved August 7, 2021, from https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/travel-during-coronavirus-pandemic/mobile-contact-tracing-apps-eu-member-states_en

Mobile Operating System Market Share Worldwide | Statcounter Global Stats. (n.d.). StatCounter Global Stats. Retrieved August 26, 2021, from <https://gs.statcounter.com/os-market-share/mobile/worldwide>

Networking with the Fast Android Networking Library | CodePath Android Cliffnotes. (n.d.). Codepath. Retrieved August 7, 2021, from <https://guides.codepath.com/android/Networking-with-the-Fast-Android-Networking-Library>

Privacy-Preserving Contact Tracing - Apple and Google. (n.d.). Apple. Retrieved August 26, 2021, from <https://covid19.apple.com/contacttracing>

Quick Start | Charts |. (n.d.). Google Developers. Retrieved August 7, 2021, from https://developers.google.com/chart/interactive/docs/quick_start
Rahman, M. (2021, February 25). *Here are the countries using Google and Apple's COVID-19 Contact Tracing API*. Xda-Developers. <https://www.xda-developers.com/google-apple-covid-19-contact-tracing-exposure-notifications-api-app-list-countries/>

Retrofit. (n.d.). Github. Retrieved August 6, 2021, from <https://square.github.io/retrofit/>

Shreevatsa. (2017, August 26). *How to handle QueryException for duplicate entries in Laravel*. Stack Overflow. <https://stackoverflow.com/questions/45893359/how-to-handle-queryexception-for-duplicate-entries-in-laravel>

The Keyword. (n.d.). https://www.blog.google/Documents/66/Overview_of_COVID-19_Contact_Tracing_Using_BLE_1.Pdf. Retrieved August 7, 2021, from <https://www.blog.google>

Wooding. (2016, December 10). *Android Studio: Create a Multiple Choice Quiz*. YouTube. https://www.youtube.com/watch?v=4g1_UH_6VQc