|  |  |  |
| --- | --- | --- |
| ΟΝΟΜΑΤΕΠΩΝΥΜΟ | ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ | EMAIL |
| Αναστοπούλου Δέσποινα | E18007 | despoinaanastopoulou@gmail.com |
| Καρτασίδου Σοφία | E20066 | sofiakarta2@gmail.com |

## Κεφάλαιο 1

## Συνοπτική περιγραφή της εργασίας

Η εφαρμογή eShop αποτελεί μια πλατφόρμα, η οποία προσφέρει στους χρήστες ένα ολοκληρωμένο σύστημα για τη διαχείριση των αγορών τους. Οι χρήστες έχουν τη δυνατότητα να περιηγούνται σε μια ποικιλία προϊόντων, να προσθέτουν αυτά που τους ενδιαφέρουν στο καλάθι αγορών ή στη λίστα επιθυμιών τους, να διαβάζουν και να αφήνουν κριτικές για τα προϊόντα που έχουν αγοράσει, και να ολοκληρώνουν τις αγορές τους με άνεση.

Οι κύριες λειτουργίες της εφαρμογής περιλαμβάνουν την αρχική σελίδα προϊόντων, όπου οι χρήστες μπορούν να εξερευνούν τις διαθέσιμες κατηγορίες και προϊόντα, το καλάθι αγορών για τη διαχείριση των επιλεγμένων προϊόντων, τη λίστα επιθυμιών για την παρακολούθηση και τη διαχείριση των αγαπημένων προϊόντων, καθώς και την οθόνη κριτικών όπου οι χρήστες μπορούν να διαβάζουν και να αφήνουν αξιολογήσεις για τα προϊόντα που έχουν αποκτήσει.

Η eShop προσφέρει μια ευέλικτη και φιλική προς τον χρήστη διαδικτυακή εμπειρία, επιτρέποντας στους χρήστες να απολαμβάνουν τις αγορές τους με ευκολία, με την επιλογή αλλαγής σε άλλη γλώσσα.

# Κεφάλαιο 2

## Αναλυτική Περιγραφή των Λειτουργιών της Εφαρμογής eShop

Η εφαρμογή eShop επιτρέπει στους χρήστες να περιηγηθούν σε προϊόντα, να τα προσθέσουν στο καλάθι αγορών ή στη λίστα επιθυμιών, να διαβάσουν και να αφήσουν κριτικές, και να ολοκληρώσουν τις αγορές τους. Οι κύριες δραστηριότητες της εφαρμογής περιλαμβάνουν την αρχική σελίδα προϊόντων, το καλάθι αγορών, τη λίστα επιθυμιών, και την οθόνη κριτικών.

### 1. Αρχική Σελίδα Προϊόντων

#### Λειτουργίες:

* **Προβολή Προϊόντων**: Εμφάνιση όλων των διαθέσιμων προϊόντων με το όνομα, την τιμή και μια μικρή περιγραφή.
* **Προσθήκη Προϊόντων στο Καλάθι**: Οι χρήστες μπορούν να προσθέσουν προϊόντα στο καλάθι αγορών τους πατώντας το κουμπί "Προσθήκη στο Καλάθι".
* **Προσθήκη Προϊόντων στη Λίστα Επιθυμιών**: Οι χρήστες μπορούν να προσθέσουν προϊόντα στη λίστα επιθυμιών τους πατώντας το κουμπί "Προσθήκη στη Λίστα Επιθυμιών".
* **Πλοήγηση**: Κουμπιά πλοήγησης στο κάτω μέρος της οθόνης για μετακίνηση στις άλλες κύριες δραστηριότητες της εφαρμογής.

### 2. Καλάθι Αγορών

#### Λειτουργίες:

* **Προβολή Προϊόντων στο Καλάθι**: Εμφάνιση όλων των προϊόντων που έχουν προστεθεί στο καλάθι του χρήστη.
* **Αύξηση Ποσότητας Προϊόντος**: Αύξηση της ποσότητας ενός προϊόντος στο καλάθι.
* **Μείωση Ποσότητας Προϊόντος**: Μείωση της ποσότητας ενός προϊόντος στο καλάθι.
* **Αφαίρεση Προϊόντος από το Καλάθι**: Αφαίρεση προϊόντος από το καλάθι.
* **Ενημέρωση Συνολικής Τιμής**: Δυναμική ενημέρωση της συνολικής τιμής του καλαθιού.
* **Πλοήγηση**: Κουμπιά πλοήγησης για μετακίνηση στις άλλες κύριες δραστηριότητες.

### 3. Λίστα Επιθυμιών

#### Λειτουργίες:

* **Προβολή Προϊόντων στη Λίστα Επιθυμιών**: Εμφάνιση όλων των προϊόντων που έχουν προστεθεί στη λίστα επιθυμιών του χρήστη.
* **Αφαίρεση Προϊόντος από τη Λίστα Επιθυμιών**: Αφαίρεση προϊόντος από τη λίστα επιθυμιών.
* **Προσθήκη Προϊόντος στο Καλάθι από τη Λίστα Επιθυμιών**: Προσθήκη προϊόντος στο καλάθι αγορών από τη λίστα επιθυμιών.
* **Πλοήγηση**: Κουμπιά πλοήγησης για μετακίνηση στις άλλες κύριες δραστηριότητες.

### 4. Ενότητα Κριτικών

#### Λειτουργίες:

* **Προβολή Κριτικών**: Εμφάνιση όλων των κριτικών των προϊόντων που έχει αγοράσει ο χρήστης.
* **Προσθήκη Κριτικής**: Δυνατότητα στους χρήστες να προσθέσουν κριτικές για τα προϊόντα που έχουν αγοράσει.
* **Πλοήγηση**: Κουμπιά πλοήγησης για μετακίνηση στις άλλες κύριες δραστηριότητες.

# Κεφάλαιο 3

# Αναλυτική Περιγραφή των Λειτουργιών της Κύριας Οθόνης Προϊόντων

Η κύρια δραστηριότητα **MainActivity** της εφαρμογής eShop είναι η πρώτη οθόνη που βλέπει ο χρήστης όταν ανοίγει την εφαρμογή. Περιλαμβάνει τη λίστα με τα διαθέσιμα προϊόντα και παρέχει διάφορες λειτουργίες, όπως η προσθήκη προϊόντων στο καλάθι αγορών, η προσθήκη στη λίστα επιθυμιών και η πλοήγηση σε άλλες δραστηριότητες της εφαρμογής.

**1. Προβολή Προϊόντων**

Η **MainActivity** εμφανίζει όλα τα διαθέσιμα προϊόντα. Κάθε προϊόν περιλαμβάνει το όνομα, την τιμή και μια μικρή περιγραφή. Η προβολή αυτή υλοποιείται μέσω ενός RecyclerView, το οποίο χρησιμοποιεί το adapter για να συνδέσει τα δεδομένα των προϊόντων με τα στοιχεία της λίστας.

Kotlin code

recyclerViewProducts = findViewById(R.id.recyclerViewProducts)

recyclerViewProducts.layoutManager = LinearLayoutManager(this)

productAdapter = ProductAdapter(productList, this)

recyclerViewProducts.adapter = productAdapter

**2. Προσθήκη Προϊόντων στο Καλάθι**

Οι χρήστες μπορούν να προσθέσουν προϊόντα στο καλάθι αγορών τους πατώντας το κουμπί "Προσθήκη στο Καλάθι" που βρίσκεται δίπλα σε κάθε προϊόν. Η λειτουργία αυτή προσθέτει το επιλεγμένο προϊόν στο καλάθι και ενημερώνει το περιεχόμενό του.

addToCartButton.setOnClickListener { cart.addToCart(product) Toast.makeText(this, "${product.name} προστέθηκε στο καλάθι.", Toast.LENGTH\_SHORT).show() }

**3. Προσθήκη Προϊόντων στη Λίστα Επιθυμιών**

Οι χρήστες μπορούν να προσθέσουν προϊόντα στη λίστα επιθυμιών τους πατώντας το κουμπί "Προσθήκη στη Λίστα Επιθυμιών" που βρίσκεται δίπλα σε κάθε προϊόν. Η λειτουργία αυτή προσθέτει το επιλεγμένο προϊόν στη λίστα επιθυμιών.

addToWishlistButton.setOnClickListener { wishlist.addToWishlist(product) Toast.makeText(this, "${product.name} προστέθηκε στη λίστα επιθυμιών.", Toast.LENGTH\_SHORT).show() }

**4. Πλοήγηση στις Κύριες Δραστηριότητες**

Η MainActivity παρέχει κουμπιά πλοήγησης στο κάτω μέρος της οθόνης, επιτρέποντας στους χρήστες να μετακινηθούν γρήγορα μεταξύ των διαφόρων κύριων δραστηριοτήτων της εφαρμογής, όπως το καλάθι αγορών, η λίστα επιθυμιών και η ενότητα κριτικών.

findViewById<Button>(R.id.navigation\_cart).setOnClickListener { startActivity(Intent(this, CartActivity::class.java)) } findViewById<Button>(R.id.navigation\_wishlist).setOnClickListener { startActivity(Intent(this, WishlistActivity::class.java)) } findViewById<Button>(R.id.navigation\_reviews).setOnClickListener { startActivity(Intent(this, ReviewActivity::class.java)) }

**5. Εναλλαγή Γλώσσας**

Η MainActivity περιλαμβάνει επίσης ένα κουμπί για την εναλλαγή της γλώσσας της εφαρμογής. Όταν ο χρήστης πατήσει το κουμπί, η γλώσσα της εφαρμογής αλλάζει μεταξύ Αγγλικών και Ελληνικών, και η δραστηριότητα επανεκκινείται για να εφαρμοστεί η νέα γλώσσα.

val languageButton: Button = findViewById(R.id.language\_button) languageButton.setOnClickListener { val newLanguage = if (localeManager.language == "en") "el" else "en" localeManager.language = newLanguage localeManager.setLocale(newLanguage) recreate() }

### Λειτουργίες ProductAdapter

Ο ProductAdapter είναι υπεύθυνος για τη σύνδεση των δεδομένων των προϊόντων με τα στοιχεία της λίστας στην κύρια δραστηριότητα (MainActivity). Χρησιμοποιείται για την προβολή των διαθέσιμων προϊόντων σε ένα RecyclerView.

#### 1. Δημιουργία ViewHolder

Η μέθοδος onCreateViewHolder δημιουργεί ένα νέο αντικείμενο ProductViewHolder για να εμφανίσει τα δεδομένα ενός προϊόντος. Το αντικείμενο αυτό περιλαμβάνει τις προβολές (views) που θα χρησιμοποιηθούν για την απεικόνιση των χαρακτηριστικών του προϊόντος.

Kotlin code

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ProductViewHolder {

val itemView = LayoutInflater.from(parent.context).inflate(R.layout.product\_item, parent, false)

return ProductViewHolder(itemView)

}

#### 2. Σύνδεση Δεδομένων με ViewHolder

Η μέθοδος onBindViewHolder συνδέει τα δεδομένα ενός προϊόντος με το ProductViewHolder. Αυτή η μέθοδος καλείται για κάθε προϊόν στη λίστα και ρυθμίζει τις προβολές με τα κατάλληλα δεδομένα.

Kotlin code

@SuppressLint("StringFormatMatches")

override fun onBindViewHolder(holder: ProductViewHolder, position: Int) {

val product = filteredList[position]

holder.productName.text = holder.itemView.context.getString(R.string.name, product.name)

holder.productDescription.text = holder.itemView.context.getString(R.string.description, product.description)

holder.productPrice.text = holder.itemView.context.getString(R.string.price, product.price)

holder.productRating.text = holder.itemView.context.getString(R.string.rating, product.rating)

holder.productDiscount.text = holder.itemView.context.getString(R.string.discount, product.discount)

holder.productCategory.text = holder.itemView.context.getString(R.string.category, product.category)

holder.productSpecifications.text = holder.itemView.context.getString(R.string.specifications, product.specifications)

holder.productSize.text = holder.itemView.context.getString(R.string.size, product.size)

holder.productColor.text = holder.itemView.context.getString(R.string.color, product.color)

holder.productAvailability.text = if (product.availability) {

holder.itemView.context.getString(R.string.availability\_in\_stock)

} else {

holder.itemView.context.getString(R.string.availability\_out\_of\_stock)

}

Glide.with(holder.productImage.context)

.load(product.imageUrl)

.into(holder.productImage)

holder.addToCartButton.setOnClickListener {

if (product.availability) {

Lists.cart.addToCart(product)

updateCallback(filteredList)

Toast.makeText(holder.itemView.context, holder.itemView.context.getString(R.string.product\_added\_to\_cart), Toast.LENGTH\_SHORT).show()

} else {

Toast.makeText(holder.itemView.context, holder.itemView.context.getString(R.string.product\_not\_available), Toast.LENGTH\_SHORT).show()

}

}

holder.addToWishlistButton.setOnClickListener {

Lists.wishlist.addToWishlist(product)

updateCallback(filteredList)

Toast.makeText(holder.itemView.context, holder.itemView.context.getString(R.string.product\_added\_to\_wishlist), Toast.LENGTH\_SHORT).show()

}

}

#### 3. Αριθμός Στοιχείων στη Λίστα

Η μέθοδος getItemCount επιστρέφει τον αριθμό των προϊόντων στη λίστα. Αυτή η μέθοδος χρησιμοποιείται από το RecyclerView για να καθορίσει πόσα στοιχεία θα εμφανιστούν.

Kotlin

override fun getItemCount(): Int {

return filteredList.size

}

### ProductViewHolder

Ο ProductViewHolder είναι μια εσωτερική κλάση που περιλαμβάνει τις προβολές (views) που χρησιμοποιούνται για την απεικόνιση των χαρακτηριστικών ενός προϊόντος. Αυτές οι προβολές περιλαμβάνουν το όνομα, την περιγραφή, την τιμή, τη βαθμολογία, την έκπτωση, την κατηγορία, τις προδιαγραφές, το μέγεθος, το χρώμα και τη διαθεσιμότητα του προϊόντος, καθώς και κουμπιά για την προσθήκη στο καλάθι και στη λίστα επιθυμιών.

Kotlin

inner class ProductViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

val productImage: ImageView = itemView.findViewById(R.id.product\_image)

val productName: TextView = itemView.findViewById(R.id.product\_name)

val productDescription: TextView = itemView.findViewById(R.id.product\_description)

val productPrice: TextView = itemView.findViewById(R.id.product\_price)

val productRating: TextView = itemView.findViewById(R.id.product\_rating)

val productDiscount: TextView = itemView.findViewById(R.id.product\_discount)

val productCategory: TextView = itemView.findViewById(R.id.product\_category)

val productSpecifications: TextView = itemView.findViewById(R.id.product\_specifications)

val productSize: TextView = itemView.findViewById(R.id.product\_size)

val productColor: TextView = itemView.findViewById(R.id.product\_color)

val productAvailability: TextView = itemView.findViewById(R.id.product\_availability)

val addToCartButton: Button = itemView.findViewById(R.id.button\_add\_to\_cart)

val addToWishlistButton: Button = itemView.findViewById(R.id.button\_add\_to\_wishlist)

}

### Ενημέρωση και Φιλτράρισμα Λίστας

Ο ProductAdapter περιλαμβάνει μεθόδους για την ενημέρωση της λίστας των προϊόντων και το φιλτράρισμα της λίστας με βάση ένα κριτήριο αναζήτησης.

#### Ενημέρωση Λίστας Προϊόντων

Η μέθοδος updateProducts ενημερώνει τη λίστα των προϊόντων και ειδοποιεί το RecyclerView για τις αλλαγές.

Kotlin

fun updateProducts(newList: List<Product>) {

filteredList = newList

notifyDataSetChanged()

}

#### Φιλτράρισμα Λίστας Προϊόντων

Η μέθοδος filter φιλτράρει τη λίστα των προϊόντων με βάση ένα κριτήριο αναζήτησης. Εάν το κριτήριο είναι κενό, επιστρέφει όλα τα προϊόντα.

Kotlin

fun filter(query: String) {

filteredList = if (query.isEmpty()) {

productList

} else {

productList.filter { product ->

product.name.contains(query, ignoreCase = true)

}

}

notifyDataSetChanged()

}

# Λειτουργίες Καλαθιού

Η δραστηριότητα **CartActivity** στην εφαρμογή eShop παρέχει στους χρήστες μια σειρά από λειτουργίες για τη διαχείριση των προϊόντων στο καλάθι αγορών τους. Η δραστηριότητα αυτή περιλαμβάνει την προβολή των προϊόντων στο καλάθι, την επεξεργασία της ποσότητας κάθε προϊόντος, την αφαίρεση προϊόντων από το καλάθι και την ενημέρωση της συνολικής τιμής. Η διαχείριση του καλαθιού γίνεται με τη βοήθεια του **CartAdapter**, που συνδέει τα δεδομένα του καλαθιού με το RecyclerView για την προβολή των προϊόντων.

## Αναλυτική Περιγραφή των Λειτουργιών

**1. Προβολή Προϊόντων στο Καλάθι**

Η **CartActivity** εμφανίζει όλα τα προϊόντα που έχουν προστεθεί στο καλάθι του χρήστη.Κάθε στοιχείο του καλαθιού περιλαμβάνει το όνομα του προϊόντος, την τιμή, την ποσότητα και κουμπιά για την αύξηση ή μείωση της ποσότητας καθώς και την αφαίρεση του προϊόντος από το καλάθι.

Kotlin code

override fun onBindViewHolder(holder: CartViewHolder, position: Int) {

val product = cartProducts.keys.toList()[position]

val quantity = cartProducts[product] ?: 0

holder.bind(product, quantity)

}

**2. Αύξηση Ποσότητας Προϊόντος**

Οι χρήστες μπορούν να αυξήσουν την ποσότητα ενός προϊόντος στο καλάθι τους, χρησιμοποιώντας το κουμπί αύξησης (increaseQuantityButton). Η ποσότητα αυξάνεται κατά 1, μέχρι ένα μέγιστο όριο 10.

Kotlin code

increaseQuantityButton.setOnClickListener {

if (quantity < 10) {

cart.updateQuantity(product, quantity + 1)

quantityTextView.text = (quantity + 1).toString()

productPriceTextView.text = itemView.context.getString(R.string.price\_format, product.price \* (quantity + 1))

updateTotalPrice()

startCartActivity() // Restart Activity

}

}

**3. Μείωση Ποσότητας Προϊόντος**

Οι χρήστες μπορούν να μειώσουν την ποσότητα ενός προϊόντος στο καλάθι τους, χρησιμοποιώντας το κουμπί μείωσης (decreaseQuantityButton). Η ποσότητα μειώνεται κατά 1, μέχρι ένα ελάχιστο όριο 1.

Kotlin code

decreaseQuantityButton.setOnClickListener {

if (quantity > 1) {

cart.updateQuantity(product, quantity - 1)

quantityTextView.text = (quantity - 1).toString()

productPriceTextView.text = itemView.context.getString(R.string.price\_format, product.price \* (quantity - 1))

updateTotalPrice()

startCartActivity() // Restart Activity

}

}

**4. Αφαίρεση Προϊόντος από το Καλάθι**

Οι χρήστες μπορούν να αφαιρέσουν προϊόντα από το καλάθι τους, χρησιμοποιώντας το κουμπί αφαίρεσης (removeFromCartButton). Το προϊόν αφαιρείται από το καλάθι και ενημερώνεται η συνολική τιμή.

Kotlin code

removeFromCartButton.setOnClickListener {

cart.removeFromCart(product)

notifyItemRemoved(adapterPosition)

notifyItemRangeChanged(adapterPosition, itemCount)

updateTotalPrice()

startCartActivity() // Restart Activity

}

**5. Ενημέρωση Συνολικής Τιμής**

Η συνολική τιμή του καλαθιού ενημερώνεται δυναμικά κάθε φορά που αλλάζει η ποσότητα ενός προϊόντος ή αφαιρείται ένα προϊόν. Η ενημέρωση αυτή γίνεται μέσω της μεθόδου updateTotalPrice().

Kotlin code

override fun onBindViewHolder(holder: CartViewHolder, position: Int) {

val product = cartProducts.keys.toList()[position]

val quantity = cartProducts[product] ?: 0

holder.bind(product, quantity)

}

private fun updateTotalPrice() {

var totalPrice = 0.0

for ((product, quantity) in cartProducts) {

totalPrice += product.price \* quantity

}

val totalPriceTextView: TextView = (itemView.context as CartActivity).findViewById(R.id.textViewTotalPrice)

val totalPriceText = itemView.context.getString(R.string.total\_price\_label, totalPrice)

totalPriceTextView.text = totalPriceText

}

**6. Πλοήγηση**

Η **CartActivity** παρέχει κουμπιά πλοήγησης στο κάτω μέρος της οθόνης για γρήγορη μετακίνηση μεταξύ των διαφόρων κύριων δραστηριοτήτων της εφαρμογής (Προϊόντα, Καλάθι, Λίστα Επιθυμιών, Κριτικές).

Kotlin code

findViewById<Button>(R.id.navigation\_products).setOnClickListener {

startActivity(Intent(this, MainActivity::class.java))

finish()

}

findViewById<Button>(R.id.navigation\_cart).setOnClickListener {

// Δεν απαιτείται ενέργεια γιατί είμαστε ήδη στο CartActivity

}

findViewById<Button>(R.id.navigation\_wishlist).setOnClickListener {

startActivity(Intent(this, WishlistActivity::class.java))

finish()

}

findViewById<Button>(R.id.navigation\_reviews).setOnClickListener {

// Κύλιση στην κορυφή της λίστας κριτικών, αν υπάρχει

recyclerViewCart.scrollToPosition(0)

}

## Αναλυτική Περιγραφή του CartAdapter

Ο CartAdapter είναι υπεύθυνος για τη διαχείριση των δεδομένων και των προβολών των προϊόντων που βρίσκονται στο καλάθι αγορών στην εφαρμογή eShop. Αυτός ο adapter συνδέει τα δεδομένα των προϊόντων του καλαθιού με τα στοιχεία του RecyclerView, επιτρέποντας στους χρήστες να βλέπουν, να επεξεργάζονται τις ποσότητες και να αφαιρούν προϊόντα από το καλάθι.

### Δομή και Λειτουργίες του CartAdapter

#### 1. Δημιουργία ViewHolder

Η μέθοδος onCreateViewHolder δημιουργεί ένα νέο αντικείμενο CartViewHolder, το οποίο περιλαμβάνει τις προβολές (views) που θα χρησιμοποιηθούν για την απεικόνιση των προϊόντων στο καλάθι.

Kotlin code

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CartViewHolder {

val inflater = LayoutInflater.from(parent.context)

val view = inflater.inflate(R.layout.cart\_item, parent, false)

return CartViewHolder(view)

}

#### 2. Σύνδεση Δεδομένων με ViewHolder

Η μέθοδος onBindViewHolder συνδέει τα δεδομένα ενός προϊόντος με το CartViewHolder. Καλείται για κάθε προϊόν στο καλάθι και ρυθμίζει τις προβολές με τα κατάλληλα δεδομένα, όπως η εικόνα του προϊόντος, το όνομα, η τιμή και η ποσότητα.

Kotlin code

override fun onBindViewHolder(holder: CartViewHolder, position: Int) {

val product = cartProducts.keys.toList()[position]

val quantity = cartProducts[product] ?: 0

holder.bind(product, quantity)

Glide.with(holder.productImage.context)

.load(product.imageUrl)

.into(holder.productImage)

}

#### 3. Αριθμός Στοιχείων στη Λίστα

Η μέθοδος getItemCount επιστρέφει τον αριθμό των προϊόντων στο καλάθι.

Kotlin code

override fun getItemCount(): Int {

return cartProducts.size

}

### Ενημέρωση Καλαθιού

Ο CartAdapter περιλαμβάνει μεθόδους για την ενημέρωση της λίστας των προϊόντων στο καλάθι και την ένδειξη εάν το καλάθι είναι άδειο.

#### Ενημέρωση Καλαθιού

Η μέθοδος updateCart ενημερώνει τη λίστα των προϊόντων στο καλάθι και ειδοποιεί το RecyclerView για τις αλλαγές.

Kotlin code

fun updateCart(newCartProducts: Map<Product, Int>) {

cartProducts = newCartProducts

notifyDataSetChanged()

}

#### Έλεγχος Αν το Καλάθι Είναι Άδειο

Η μέθοδος isEmpty επιστρέφει true αν το καλάθι είναι άδειο.

Kotlin code

fun isEmpty(): Boolean {

return cartProducts.isEmpty()

}

### CartViewHolder

Ο CartViewHolder είναι μια εσωτερική κλάση που περιλαμβάνει τις προβολές (views) που χρησιμοποιούνται για την απεικόνιση των χαρακτηριστικών ενός προϊόντος στο καλάθι. Περιλαμβάνει επίσης λειτουργίες για την αύξηση/μείωση της ποσότητας και την αφαίρεση του προϊόντος από το καλάθι.

Kotlin

inner class CartViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

val productImage: ImageView = itemView.findViewById(R.id.product\_image)

private val productNameTextView: TextView = itemView.findViewById(R.id.productNameTextView)

private val productPriceTextView: TextView = itemView.findViewById(R.id.productPriceTextView)

private val quantityTextView: TextView = itemView.findViewById(R.id.quantityTextView)

private val increaseQuantityButton: Button = itemView.findViewById(R.id.increaseQuantityButton)

private val decreaseQuantityButton: Button = itemView.findViewById(R.id.decreaseQuantityButton)

private val removeFromCartButton: Button = itemView.findViewById(R.id.removeFromCartButton)

fun bind(product: Product, quantity: Int) {

productNameTextView.text = product.name

productPriceTextView.text = itemView.context.getString(R.string.price\_format, product.price \* quantity)

quantityTextView.text = quantity.toString()

increaseQuantityButton.setOnClickListener {

if (quantity < 10) {

cart.updateQuantity(product, quantity + 1)

quantityTextView.text = (quantity + 1).toString()

productPriceTextView.text = itemView.context.getString(R.string.price\_format, product.price \* (quantity + 1))

updateTotalPrice()

startCartActivity() // Restart Activity

}

}

decreaseQuantityButton.setOnClickListener {

if (quantity > 1) {

cart.updateQuantity(product, quantity - 1)

quantityTextView.text = (quantity - 1).toString()

productPriceTextView.text = itemView.context.getString(R.string.price\_format, product.price \* (quantity - 1))

updateTotalPrice()

startCartActivity() // Restart Activity

}

}

removeFromCartButton.setOnClickListener {

cart.removeFromCart(product)

notifyItemRemoved(adapterPosition)

notifyItemRangeChanged(adapterPosition, itemCount)

updateTotalPrice()

startCartActivity() // Restart Activity

}

}

private fun updateTotalPrice() {

var totalPrice = 0.0

for ((product, quantity) in cartProducts) {

totalPrice += product.price \* quantity

}

val totalPriceTextView: TextView = (itemView.context as CartActivity).findViewById(R.id.textViewTotalPrice)

val totalPriceText = itemView.context.getString(R.string.total\_price\_label, totalPrice)

totalPriceTextView.text = totalPriceText

}

private fun startCartActivity() {

val intent = Intent(context, CartActivity::class.java)

context.startActivity(intent)

}

}

# Λειτουργίες Λίστας Επιθυμιών

Η Wishlist παρέχει στους χρήστες μια σειρά από λειτουργίες για τη διαχείριση των προϊόντων στη λίστα επιθυμιών τους. Η δραστηριότητα αυτή περιλαμβάνει την προβολή των προϊόντων στη λίστα επιθυμιών, την αφαίρεση προϊόντων από αυτή και την προσθήκη προϊόντων στο καλάθι.

### Αναλυτική Περιγραφή των Λειτουργιών

**1. Προβολή Προϊόντων στη Λίστα Επιθυμιών**

Η WishlistActivity εμφανίζει όλα τα προϊόντα που έχουν προστεθεί στη λίστα επιθυμιών του χρήστη. Αυτή η λειτουργία υλοποιείται μέσω του RecyclerView και του WishlistAdapter. Κάθε στοιχείο της λίστας περιλαμβάνει το όνομα του προϊόντος, την τιμή, κουμπί για την αφαίρεση του προϊόντος από τη λίστα επιθυμιών και κουμπί για την προσθήκη του προϊόντος στο καλάθι.

Kotlin code

recyclerViewWishlist = findViewById(R.id.recyclerViewWishlist)

recyclerViewWishlist.layoutManager = LinearLayoutManager(this)

recyclerViewWishlist.adapter = WishlistAdapter(wishlistProducts, wishlist, this)

**2. Αφαίρεση Προϊόντος από τη Λίστα Επιθυμιών**

Οι χρήστες μπορούν να αφαιρέσουν προϊόντα από τη λίστα επιθυμιών τους, χρησιμοποιώντας το κουμπί αφαίρεσης (removeFromWishlistButton). Το προϊόν αφαιρείται από τη λίστα και η προβολή ενημερώνεται κατάλληλα.

Kotlin code

removeFromWishlistButton.setOnClickListener {

wishlist.removeFromWishlist(product.id)

notifyItemRemoved(adapterPosition)

notifyItemRangeChanged(adapterPosition, itemCount)

}

**3.Προσθήκη Προϊόντος στο Καλάθι**

Οι χρήστες μπορούν να προσθέσουν προϊόντα από τη λίστα επιθυμιών τους στο καλάθι αγορών, χρησιμοποιώντας το κουμπί προσθήκης στο καλάθι (addToCartButton). Το προϊόν προστίθεται στο καλάθι και αφαιρείται από τη λίστα επιθυμιών.

Kotlin code

addToCartButton.setOnClickListener {

cart.addToCart(product)

wishlist.removeFromWishlist(product.id)

notifyItemRemoved(adapterPosition)

notifyItemRangeChanged(adapterPosition, itemCount)

}

**4. Πλοήγηση**

Η WishlistActivity παρέχει κουμπιά πλοήγησης στο κάτω μέρος της οθόνης για γρήγορη μετακίνηση μεταξύ των διαφόρων κύριων δραστηριοτήτων της εφαρμογής (Προϊόντα, Καλάθι, Λίστα Επιθυμιών, Κριτικές).

Kotlin code

findViewById<Button>(R.id.navigation\_products).setOnClickListener {

startActivity(Intent(this, MainActivity::class.java))

finish()

}

findViewById<Button>(R.id.navigation\_cart).setOnClickListener {

startActivity(Intent(this, CartActivity::class.java))

finish()

}

findViewById<Button>(R.id.navigation\_wishlist).setOnClickListener {

// Δεν απαιτείται ενέργεια γιατί είμαστε ήδη στο WishlistActivity

}

findViewById<Button>(R.id.navigation\_reviews).setOnClickListener {

startActivity(Intent(this, ReviewsActivity::class.java))

finish()

}

## Αναλυτική Περιγραφή του WishlistAdapter

Ο WishlistAdapter είναι υπεύθυνος για τη διαχείριση των δεδομένων και των προβολών των προϊόντων που βρίσκονται στη λίστα επιθυμιών στην εφαρμογή eShop. Αυτός ο adapter συνδέει τα δεδομένα των προϊόντων της λίστας επιθυμιών με τα στοιχεία του RecyclerView, επιτρέποντας στους χρήστες να βλέπουν, να προσθέτουν προϊόντα στο καλάθι και να αφαιρούν προϊόντα από τη λίστα επιθυμιών.

### Δομή και Λειτουργίες του WishlistAdapter

#### 1. Δημιουργία ViewHolder

Η μέθοδος onCreateViewHolder δημιουργεί ένα νέο αντικείμενο WishlistViewHolder, το οποίο περιλαμβάνει τις προβολές (views) που θα χρησιμοποιηθούν για την απεικόνιση των προϊόντων στη λίστα επιθυμιών.

Kotlin code

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): WishlistViewHolder {

val itemView = LayoutInflater.from(parent.context).inflate(R.layout.wishlist\_item, parent, false)

return WishlistViewHolder(itemView)

}

#### 2. Σύνδεση Δεδομένων με ViewHolder

Η μέθοδος onBindViewHolder συνδέει τα δεδομένα ενός προϊόντος με το WishlistViewHolder. Καλείται για κάθε προϊόν στη λίστα επιθυμιών και ρυθμίζει τις προβολές με τα κατάλληλα δεδομένα, όπως η εικόνα του προϊόντος, το όνομα και η τιμή.

Kotlin code

override fun onBindViewHolder(holder: WishlistViewHolder, position: Int) {

val productList = wishlistProducts.toList() // Convert set to list

val product = productList[position]

holder.bind(product)

Glide.with(holder.productImage.context)

.load(product.imageUrl)

.into(holder.productImage)

}

#### 3. Αριθμός Στοιχείων στη Λίστα

Η μέθοδος getItemCount επιστρέφει τον αριθμό των προϊόντων στη λίστα επιθυμιών.

Kotlin code

override fun getItemCount(): Int {

return wishlistProducts.size

}

### Ενημέρωση Λίστας Επιθυμιών

Ο WishlistAdapter περιλαμβάνει μεθόδους για την ενημέρωση της λίστας επιθυμιών και την ένδειξη εάν η λίστα είναι άδεια.

#### Ενημέρωση Λίστας Επιθυμιών

Η μέθοδος updateWishlist ενημερώνει τη λίστα των προϊόντων στη λίστα επιθυμιών και ειδοποιεί το RecyclerView για τις αλλαγές.

Kotlin code

fun updateWishlist(newWishlist: List<Product>) {

wishlistProducts = newWishlist.toSet()

notifyDataSetChanged()

}

#### Έλεγχος Αν η Λίστα Επιθυμιών Είναι Άδεια

Η μέθοδος isEmpty επιστρέφει true αν η λίστα επιθυμιών είναι άδεια.

Kotlin code

fun isEmpty(): Boolean {

return wishlistProducts.isEmpty()

}

### WishlistViewHolder

Ο WishlistViewHolder είναι μια εσωτερική κλάση που περιλαμβάνει τις προβολές (views) που χρησιμοποιούνται για την απεικόνιση των χαρακτηριστικών ενός προϊόντος στη λίστα επιθυμιών. Περιλαμβάνει επίσης λειτουργίες για την αφαίρεση του προϊόντος από τη λίστα επιθυμιών και την προσθήκη του στο καλάθι αγορών.

Kotlin code

inner class WishlistViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

val productImage: ImageView = itemView.findViewById(R.id.product\_image)

private val productNameTextView: TextView = itemView.findViewById(R.id.product\_name)

private val productPriceTextView: TextView = itemView.findViewById(R.id.product\_price)

private val removeFromWishlistButton: Button = itemView.findViewById(R.id.button\_remove\_from\_wishlist)

private val addToCartButton: Button = itemView.findViewById(R.id.button\_add\_to\_cart)

fun bind(product: Product) {

productNameTextView.text = product.name

productPriceTextView.text = itemView.context.getString(R.string.price\_format, product.price)

removeFromWishlistButton.setOnClickListener {

wishlist.removeFromWishlist(product)

notifyItemRemoved(adapterPosition)

notifyItemRangeChanged(adapterPosition, itemCount)

startWishlistActivity() // Restart Activity

}

addToCartButton.setOnClickListener {

if (product.availability) {

cart.addToCart(product)

wishlist.removeFromWishlist(product)

notifyItemRemoved(adapterPosition)

notifyItemRangeChanged(adapterPosition, itemCount)

startWishlistActivity() // Restart Activity

} else {

Toast.makeText(itemView.context, itemView.context.getString(R.string.product\_not\_available), Toast.LENGTH\_SHORT).show()

}

}

}

private fun startWishlistActivity() {

val intent = Intent(context, WishlistActivity::class.java)

context.startActivity(intent)

}

}

## Αναλυτική Περιγραφή του ReviewActivity

Η κλάση ReviewActivity αποτελεί μια δραστηριότητα (Activity) στην εφαρμογή σας, η οποία διαχειρίζεται την εμφάνιση και τη λειτουργικότητα για τις κριτικές προϊόντων.

#### Μεταβλητές κλάσης και αρχικοποίηση

kotlin

class ReviewActivity : AppCompatActivity() {

private lateinit var reviewRecyclerView: RecyclerView

private lateinit var reviewAdapter: ReviewAdapter

private lateinit var boughtProducts: BoughtProducts

private lateinit var emptyMessage: TextView

// Locale Manager

private lateinit var localeManager: LocaleManager

* reviewRecyclerView: Το RecyclerView που χρησιμοποιείται για την εμφάνιση των κριτικών προϊόντων.
* reviewAdapter: Ο Adapter που συνδέει τα δεδομένα με το RecyclerView.
* boughtProducts: Το αντικείμενο που περιέχει τα αγορασμένα προϊόντα.
* emptyMessage: Το TextView που εμφανίζει ένα μήνυμα όταν η λίστα κριτικών είναι άδεια.
* localeManager: Ο διαχειριστής της γλώσσας για την εφαρμογή.

#### Μέθοδος onCreate

Kotlin

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

// Initialize Locale Manager

localeManager = LocaleManager(this)

// Set the language to the saved language on startup

localeManager.language?.let { localeManager.setLocale(it) }

setContentView(R.layout.activity\_review)

val toolbar: Toolbar = findViewById(R.id.toolbar)

setSupportActionBar(toolbar)

supportActionBar?.setDisplayShowTitleEnabled(false)

// Initialize boughtProducts

boughtProducts = Lists.boughtProducts

// Initialize RecyclerView and Adapter

reviewRecyclerView = findViewById(R.id.recyclerViewReviews)

reviewRecyclerView.layoutManager = LinearLayoutManager(this)

reviewAdapter = ReviewAdapter(boughtProducts.getBoughtProducts(), boughtProducts, this)

reviewRecyclerView.adapter = reviewAdapter

// Initialize empty message view

emptyMessage = findViewById(R.id.empty\_review)

// Check and update visibility of empty message

checkReviewEmpty()

// Fetch bought products from Firebase and update the adapter

fetchBoughtProducts()

// Set click listeners for bottom navigation buttons

setupBottomNavigation()

// Language button setup

setupLanguageButton()

}

* Η **onCreate** καλείται κατά τη δημιουργία της δραστηριότητας. Εδώ αρχικοποιείται ο **LocaleManager** για τη γλώσσα της εφαρμογής, και ορίζεται η γλώσσα από τις αποθηκευμένες ρυθμίσεις. Το **layout** της δραστηριότητας ορίζεται στο **activity\_review.xml.**
* Αρχικοποιείται το αντικείμενο **boughtProducts** που περιέχει τα αγορασμένα προϊόντα του χρήστη.
* Το r**eviewRecyclerView** δημιουργείται και ρυθμίζεται με έναν **LinearLayoutManager** για τη διαχείριση της διάταξης των αντικειμένων.
* Ο **reviewAdapter** αρχικοποιείται με τα αγορασμένα προϊόντα και τον περνάμε στο **reviewRecyclerView** ως **adapter**.
* Το **emptyMessage αρχικοποιείται** για την εμφάνιση μηνύματος όταν δεν υπάρχουν κριτικές προϊόντων.
* Η μέθοδος **checkReviewEmpty()** καλείται για να ελέγξει αν η λίστα κριτικών είναι άδεια και να ενημερώσει την οθόνη ανάλογα.
* Η **fetchBoughtProducts(**) καλείται για να φορτώσει τα αγορασμένα προϊόντα από το Firebase και να ενημερώσει τον reviewAdapter με τα νέα δεδομένα.
* Οι μέθοδοι **setupBottomNavigation()** και **setupLanguageButton()** ορίζουν τους κλικ listeners για τα κουμπιά πλοήγησης και το κουμπί γλώσσας αντίστοιχα.

## Αναλυτική Περιγραφή του ReviewAdapter

Η κλάση αυτή διαχειρίζεται την απεικόνιση των προϊόντων που έχουν αγοραστεί και την υποβολή κριτικών για αυτά τα προϊόντα.

#### 1. Κατασκευαστής και αρχικοποίηση

Kotlin code

class ReviewAdapter(

private var boughtProductsList: Set<Product>,

private val boughtProducts: BoughtProducts,

private val context: Context

) : RecyclerView.Adapter<ReviewAdapter.ReviewViewHolder>() {

// Constructor and methods

}

 **boughtProductsList**: Λίστα με τα προϊόντα που έχουν αγοραστεί και είναι διαθέσιμα για κριτική.

 **context**: Περιβάλλον για την εκκίνηση νέων δραστηριοτήτων.

#### 2. Μέθοδος onCreateViewHolder

Δημιουργεί ένα νέο ReviewViewHolder που περιλαμβάνει τη διάταξη του κάθε στοιχείου κριτικής.

Kotlin code

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ReviewViewHolder {

val view = LayoutInflater.from(parent.context).inflate(R.layout.review\_item, parent, false)

return ReviewViewHolder(view)

}

* Φορτώνει το layout review\_item που ορίζει την εμφάνιση κάθε κριτικής.

#### 3. Μέθοδος onBindViewHolder

Συνδέει τα δεδομένα ενός προϊόντος με το ReviewViewHolder και ρυθμίζει τα στοιχεία της κριτικής.

Kotlin code

override fun onBindViewHolder(holder: ReviewViewHolder, position: Int) {

val productList = boughtProductsList.toList() // Μετατροπή σε λίστα

val product = productList[position]

holder.bind(product)

}

* Επιλέγει το συγκεκριμένο προϊόν στη θέση position από τη λίστα αγορασμένων προϊόντων και το συνδέει με τον ReviewViewHolder.

#### 4. Μέθοδος updateBoughtProducts

Ενημερώνει τη λίστα των αγορασμένων προϊόντων και ειδοποιεί το RecyclerView για τις αλλαγές.

Kotlin code

fun updateBoughtProducts(newBoughtProducts: List<Product>) {

boughtProductsList = newBoughtProducts.toSet()

notifyDataSetChanged()

}

* Χρησιμοποιείται για να ενημερώσει τη λίστα των αγορασμένων προϊόντων όταν αλλάζει η λίστα στο Firebase.

#### 5. Μέθοδος getItemCount

Επιστρέφει τον αριθμό των προϊόντων που υπάρχουν στη λίστα αγορασμένων προϊόντων.

Kotlin code

override fun getItemCount(): Int {

return boughtProductsList.size

}

#### 6. Εσωτερική κλάση ReviewViewHolder

Αναλαμβάνει την αναπαράσταση και τη διαχείριση των στοιχείων κάθε κριτικής στο RecyclerView.

Kotlin code

inner class ReviewViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

val productName: TextView = itemView.findViewById(R.id.productName)

val ratingSpinner: Spinner = itemView.findViewById(R.id.ratingSpinner)

val commentEditText: EditText = itemView.findViewById(R.id.commentEditText)

val reviewButton: Button = itemView.findViewById(R.id.reviewButton)

fun bind(product: Product) {

productName.text = product.name

// Δημιουργία adapter για το spinner της αξιολόγησης

val ratingAdapter = ArrayAdapter.createFromResource(itemView.context,

R.array.rating\_array, android.R.layout.simple\_spinner\_item)

ratingAdapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)

ratingSpinner.adapter = ratingAdapter

// Ορισμός listener για το κουμπί κριτικής

reviewButton.setOnClickListener {

val rating = ratingSpinner.selectedItem.toString().toFloat()

val comment = commentEditText.text.toString().trim()

if (comment.isEmpty()) {

Toast.makeText(itemView.context, itemView.context.getString(R.string.empty\_comment), Toast.LENGTH\_SHORT).show()

return@setOnClickListener

}

// Αποθήκευση κριτικής στη βάση δεδομένων Firebase

val database = FirebaseDatabase.getInstance().reference.child("reviews")

database.addListenerForSingleValueEvent(object : ValueEventListener {

override fun onDataChange(snapshot: DataSnapshot) {

var reviewId = 1

while (snapshot.hasChild(reviewId.toString())) {

reviewId++

}

// Δημιουργία νέας κριτικής

val review = Review(reviewId, product.id, comment, rating)

database.child(reviewId.toString()).setValue(review)

// Σήμανση προϊόντος ως κριτικάρισμένο στη λίστα αγορασμένων προϊόντων

boughtProducts.removeProduct(product)

// Ενημέρωση αξιολόγησης προϊόντος

val reviews = snapshot.children.mapNotNull { it.getValue(Review::class.java) }

val updatedRating = calculateAverageRating(product, reviews, review)

updateProductRatingInDatabase(product, updatedRating)

// Ενημέρωση του adapter για τις αλλαγές στα δεδομένα

updateBoughtProducts(boughtProductsList.toList()) // Ανανέωση της λίστας αγορασμένων προϊόντων

Toast.makeText(itemView.context, itemView.context.getString(R.string.review\_posted), Toast.LENGTH\_SHORT).show()

startReviewActivity() // Επανεκκίνηση δραστηριότητας

}

override fun onCancelled(error: DatabaseError) {

// Χειρισμός σφαλμάτων

}

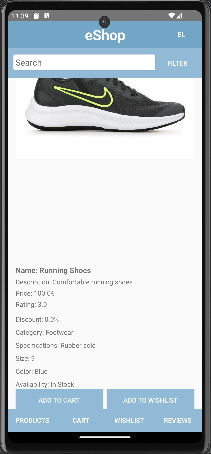
})

}

}

Κεφάλαιο 4: Εγχειρίδιο χρήστη

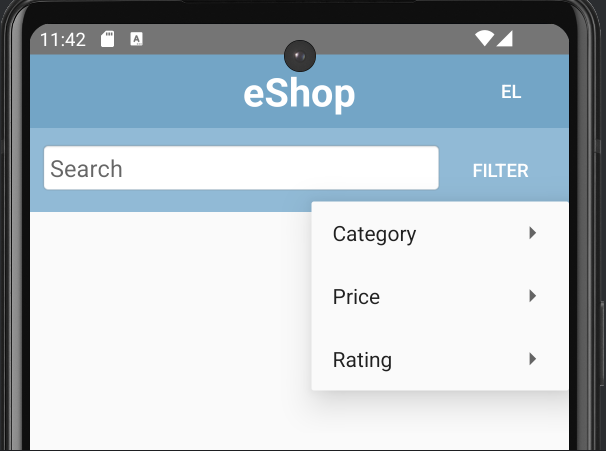
## Λίστα Προϊόντων



Η αρχική σελίδα εμφανίζει μια λίστα με όλα τα διαθέσιμα προϊόντα. Κάθε προϊόν περιλαμβάνει το όνομα, την τιμή, μια μικρή περιγραφή, το rating,το μέγεθος ,το χρώμα ,την κατηγορία και την διαθεσιμότητα.

Οι χρήστες μπορούν να προσθέσουν προϊόντα στο καλάθι τους πατώντας το κουμπί "Προσθήκη στο Καλάθι" (addToCartButton). Αυτή η λειτουργία προσθέτει το επιλεγμένο προϊόν στο καλάθι αγορών και εμφανίζει ένα μήνυμα επιβεβαίωσης.

Οι χρήστες μπορούν να προσθέσουν προϊόντα στη λίστα επιθυμιών τους πατώντας το κουμπί "Προσθήκη στη Λίστα Επιθυμιών" (addToWishlistButton). Αυτή η λειτουργία αποθηκεύει το επιλεγμένο προϊόν στη λίστα επιθυμιών και εμφανίζει ένα μήνυμα επιβεβαίωσης.



Οι χρήστες μπορούν να αναζητήσουν τα προϊόντα με φίλτρα, υπάρχουν 3 είδη, με βάση την κατηγορία, την τιμή και τέλος με βάση την αξιολόγηση.

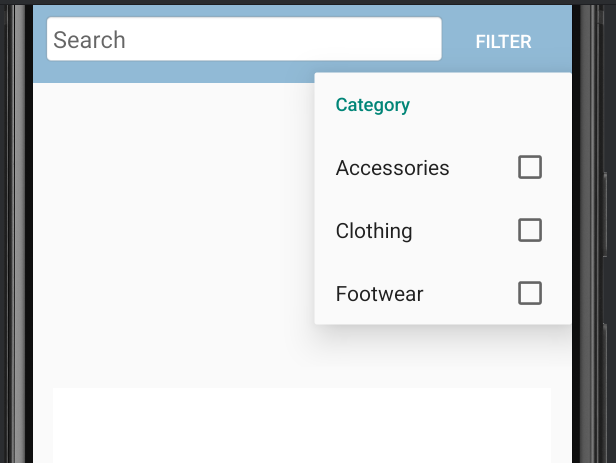
Το κάθε φίλτρο έχει επιμέρους φίλτρα

A screen shot of a computer

Description automatically generated

A screenshot of a search

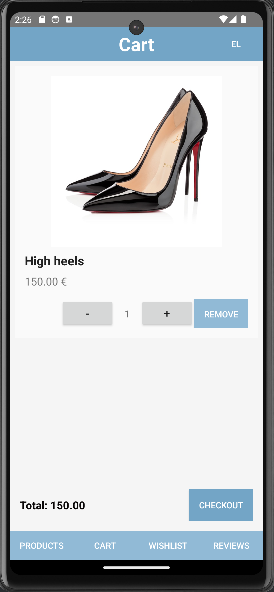
Description automatically generated



Επίσης στο κάτω μέρος της οθόνης υπάρχει το Navigation Map με το οποίο μπορεί να πλοηγηθεί στην εφαρμογή.

Το κουμπί EL δίνει την δυνατότητα αλλαγής γλώσσας στα ελληνικά.

## Καλάθι Αγορών



Αφού ο χρήστης πατήσει ADD TO CART στα προϊόντα που επιθυμεί να αγοράσει μπορεί να πατήσει την επιλογή CART στο navigation bar,η οποία θα τον κατευθύνει στο καλάθι.

A screen shot of a phone

Description automatically generated

Ο χρήστης έχει την δυνατότητα να επεξεργαστεί την ποσότητα των προϊόντων που έχει επιλέξει, να αυξήσει την ποσότητα πατώντας το (+) και να την μειώσει πατώντας (-).Επίσης έχει την δυνατότητα να τα αφαιρέσει από το καλάθι πατώντας REMOVE.

Το κουμπί EL δίνει την δυνατότητα αλλαγής γλώσσας στα ελληνικά.

A screenshot of a phone

Description automatically generated

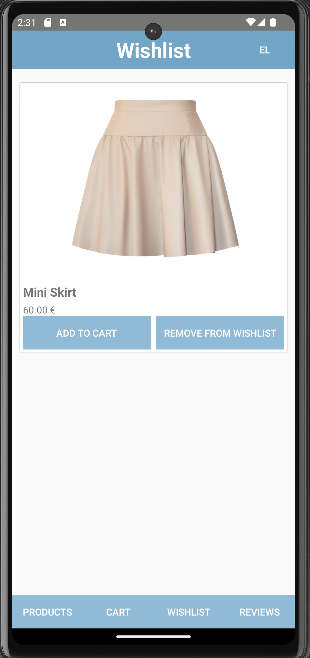
Το συνολικό πόσο εμφανίζεται κάτω αριστερά και μόλις ο χρήστης θελήσει να ολοκληρώσει την συναλλαγή μπορεί να πατήσει το κουμπί CHECKOUT.

A screenshot of a phone

Description automatically generated

Αφού η συναλλαγή εκτελεστεί με επιτυχία θα εμφανιστεί στον χρήστη σχετικό μήνυμα στο κάτω μέρος της οθόνης.

## Λίστα Επιθυμιών



Αφού ο χρήστης πατήσει ADD TO WISHLIST στα προϊόντα που επιθυμεί να βάλει στην λίστα επιθυμιών μπορεί να πατήσει την επιλογή WISHLIST στο navigation bar,η οποία θα τον κατευθύνει στη λίστα.

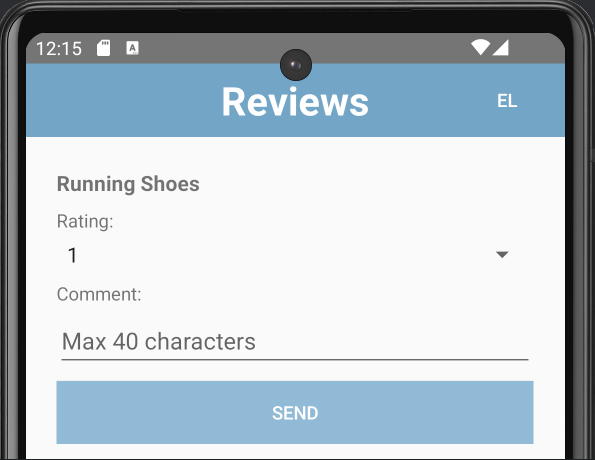
A screenshot of a phone

Description automatically generated

Ο χρήστης έχει την δυνατότητα να επεξεργαστεί την λίστα επιθυμιών, να προσθέσει το προϊόν στο καλάθι πατώντας ADD TO CART αλλά και να αφαιρέσει το προϊόν από την λίστα πατώντας το κουμπί REMOVE.

Το κουμπί EL δίνει την δυνατότητα αλλαγής γλώσσας στα ελληνικά.

## Κριτικές



Αφού ο χρήστης ολοκληρώσει την αγορά τα αγορασμένα προϊόντα εμφανίζονται στην οθόνη Reviews όπου μπορεί να αφήσει μία κριτική, η οποία αποτελείται από μια παράγραφο 40 χαρακτήρων. Πατώντας SEND υποβάλει την κρητική του.