Aplikacje Mobilne

Sprawozdanie Laboratoria

Adrian Kokot (148165), Wiktor Szymański (148084)

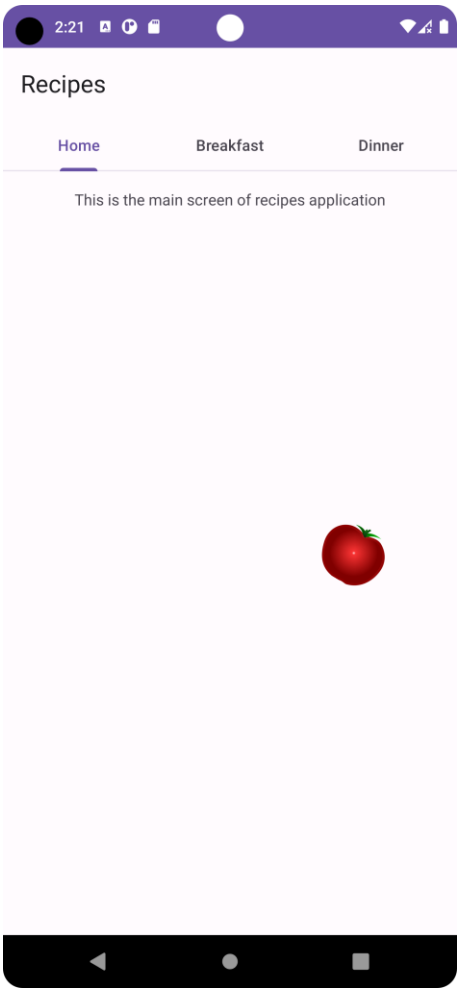
Spis treści

Aplikacja typu lista-szczegóły

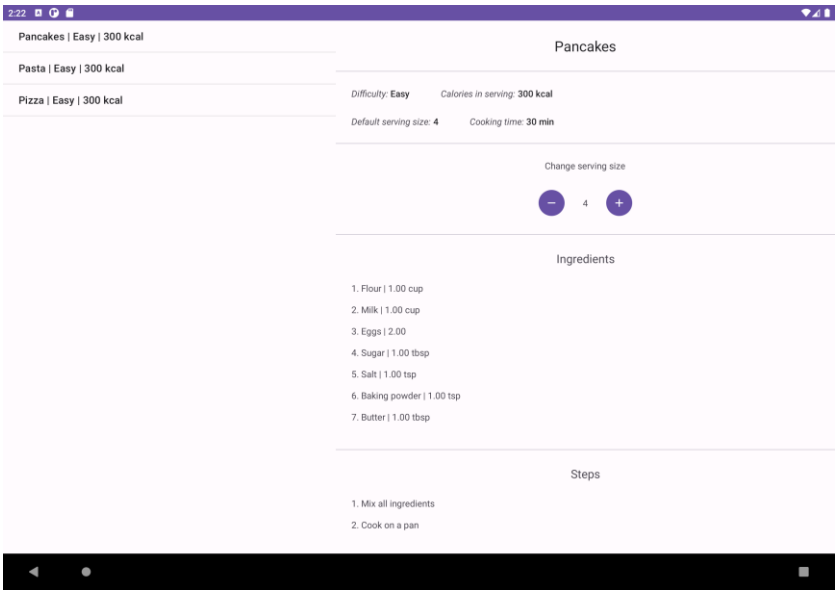
Aplikacja jest typu lista-szczegóły. Została napisana w Kotlinie. Aplikacja jest podzielona na dwie aktywności – MainActivity (lista potraw) oraz DetailActivity (szczegóły potrawy). Aplikacja ma dwie wersje układu – dla smartfonów i dla tabletów. Aplikacja korzysta z fragmentów.

```kotlin
class MainActivity : AppCompatActivity(), RecipeListFragment.Companion.Listener {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    override fun itemClicked(id: Int) {
        val fragmentContainer = findViewById<View>(R.id.fragment_container)

        if (fragmentContainer != null) {
            val transaction = supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container, RecipeDetailFragment.newInstance(id))
            transaction.addToBackStack(null)
            transaction.setTransition(androidx.fragment.app.FragmentTransaction.TRANSIT_FRAGMENT_FADE)
            transaction.commit()
        } else {
            val intent = Intent(this, DetailActivity::class.java)
            intent.putExtra(DetailActivity.RECIPE_ID, id)
            startActivity(intent)
        }
    }
}
```

Kod 1 MainActivity - wstawianie odpowiedniego fragmentu w przypadku wersji dla tabletów



Zdjęcie 1 Układ MainActivity dla smartfona

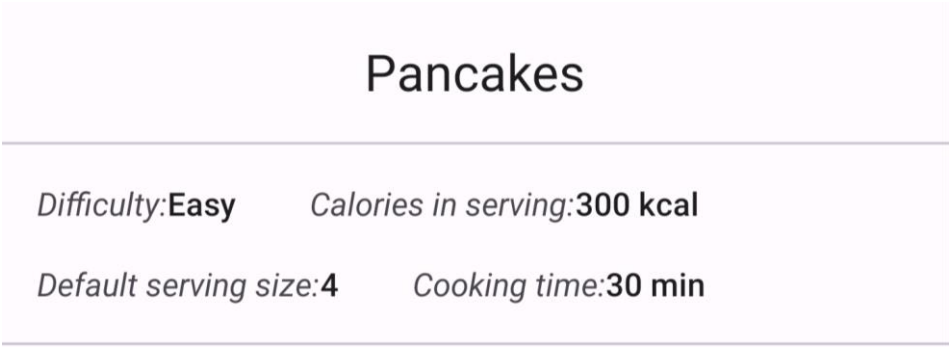


Zdjęcie 2 Układ MainActivity dla tableta

Potrawy są przechowywane w bazie danych utworzonej za pomocą biblioteki ROOM. Fragment odpowiedzialny za szczegóły potrawy sprawdza szczegóły potrawy w bazie danych.

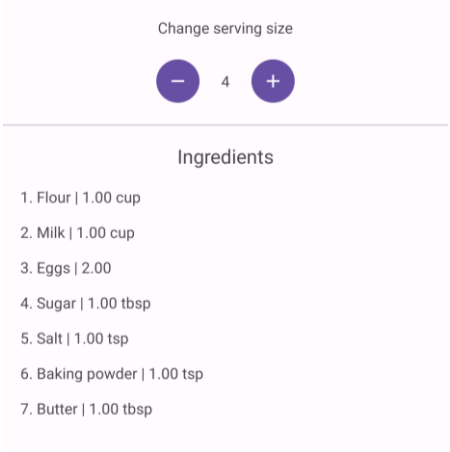
```kotlin
override fun onStart() {
    super.onStart()

    val view = this.view
    val recipeId = this.recipeId

    if (view == null || recipeId == null) {
        return
    }

    AppDatabase.getDatabase(requireContext())
        .recipeDao()
        .getRecipe(recipeId)
        .observe(viewLifecycleOwner) { recipe ->
            recipeIngredientsShare = "Recipe: ${recipe.recipe.name}\nIngredients:\n${recipe.getIngredientsString()}"
            view.findViewById<TextView>(R.id.recipe_name).text = recipe.recipe.name
            view.findViewById<TextView>(R.id.recipe_difficulty).text = recipe.recipe.difficulty
            view.findViewById<TextView>(R.id.recipe_calories).text = "${recipe.recipe.calories} kcal"
            view.findViewById<TextView>(R.id.recipe_cooking_time).text = "${recipe.recipe.cookingTime} min"
            view.findViewById<TextView>(R.id.recipe_serving_size).text = recipe.recipe.servingSize.toString()
            view.findViewById<TextView>(R.id.cooking_steps).text = recipe.getStepsString()
            view.findViewById<FloatingActionButton?>(R.id.fab)?.setOnClickListener(::onFabClick)
            view.findViewById<MaterialToolbar?>(R.id.topAppBar)?.let { toolbar ->
                toolbar.title = recipe.recipe.name
                view.findViewById<ImageView>(R.id.toolbar_recipe_image).setImageResource(recipe.recipe.imageId)
                toolbar.setNavigationOnClickListener {
                    requireActivity().finish()
                }
            }
        }
}
```

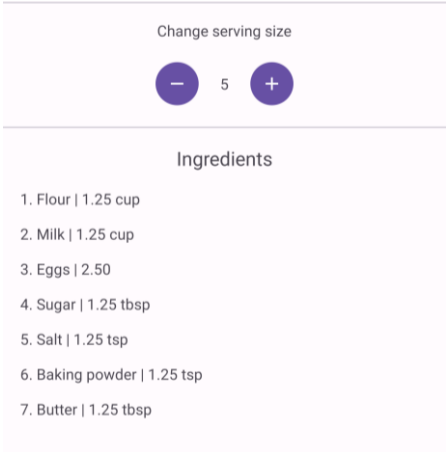
Kod 2 Wyświetlanie szczegółów potrawy w RecipeDetailFragment@onStart



Zdjęcie 3 Szczegóły potrawy

Na widoku szczegółów potrawy znajduje się także fragment odpowiedzialny za przeliczanie ilości składników w zależności od wybranej ilości porcji.





Zdjęcie 4 Przelicznik ilości składników w zależności od porcji

```kotlin
private fun renderUi() {
    val handler = Handler(Looper.getMainLooper())

    handler.post {
        val view = this.view ?: return@post
        val recipe = this.recipe ?: return@post
        val servingSize = this.servingSize ?: return@post

        view.findViewById<TextView>(R.id.ingredients).text = recipe.getIngredientsString(servingSize)
        view.findViewById<TextView>(R.id.serving_size).text = servingSize.toString()
    }
}
```

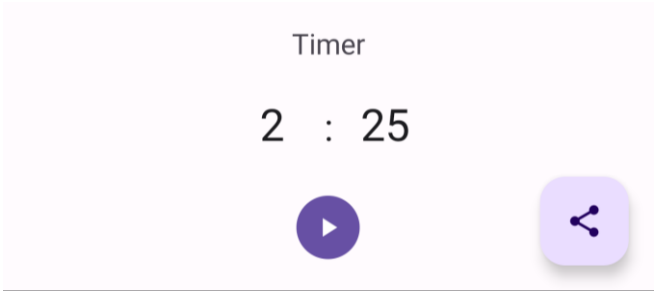
Kod 3 Obliczanie ilości składników w IngredientsFragment@renderUi

```kotlin
fun getIngredientsString(servingSize: Int = recipe.servingSize): String {
    return ingredients.mapIndexed { index, ingredient ->
        "%d. %s | %.2f %s\n\t%s".format(
            index + 1,
            ingredient.name,
            ingredient.amount * servingSize / recipe.servingSize,
            ingredient.unit,
            ingredient.comment
        )
    }.joinToString("\n")
}
```

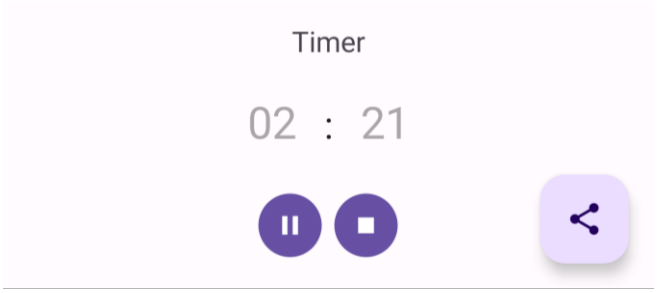
Kod 4 Generowanie tekstu ze składnikami w Recipe@getIngredientsString

Fragment dynamiczny

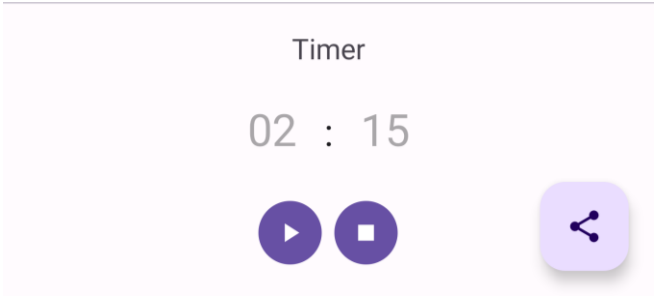
Na widoku szczegółów potrawy jest zagnieżdżony fragment z minutnikiem. Minutnik można pauzować i przerywać. Gdy minutnik się dojdzie do zera to uruchamia się dźwięk (narastający) oraz wibracje. Minutnik można ustawić przed startem.



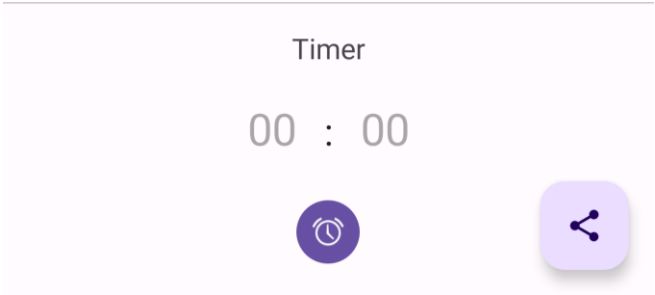
Zdjęcie 5 Edycja minutnika



Zdjęcie 6 Uruchomiony minutnik



Zdjęcie 7 Wstrzymany minutnik



Zdjęcie 8 Dzwoniący minutnik

```kotlin
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    val view = inflater.inflate(R.layout.fragment_timer, container, false)

    secondsView = view.findViewById(R.id.timer_seconds)
    minutesView = view.findViewById(R.id.timer_minutes)
    startButton = view.findViewById(R.id.start_timer_button)
    stopButton = view.findViewById(R.id.stop_timer_button)
    pauseButton = view.findViewById(R.id.pause_timer_button)
    alarmButton = view.findViewById(R.id.stop_alarm_button)

    secondsView.filters = arrayOf(MinMaxFilter(0, 59))
    minutesView.filters = arrayOf(MinMaxFilter(0, 180))

    startButton.setOnClickListener {
        if (secondsView.text.isEmpty() || minutesView.text.isEmpty()) {
            return@setOnClickListener
        }

        seconds = secondsView.text.toString().toInt() + minutesView.text.toString().toInt() * 60
        if (seconds <= 0) {
            return@setOnClickListener
        }

        running = true
        refreshUIStates()
    }

    pauseButton.setOnClickListener {
        running = false
        refreshUIStates()
    }

    stopButton.setOnClickListener {
        running = false
        seconds = 0
        secondsView.setText("%02d".format(0))
        minutesView.setText("%02d".format(0))
        refreshUIStates()
    }

    alarmButton.setOnClickListener {
        running = false
        stopAlarm()
        initVolumeShaper()
        refreshUIStates()
    }

    runTimer()
    refreshUIStates()

    return view
}


private fun refreshUIStates() {
    secondsView.isEnabled = !running && seconds == 0
    minutesView.isEnabled = !running && seconds == 0
    startButton.visibility = if (!running || (!running && seconds > 0)) View.VISIBLE else View.GONE
    pauseButton.visibility = if (running && !(this::mediaPlayer.isInitialized && mediaPlayer.isPlaying)) View.VISIBLE else View.GONE
    stopButton.visibility = if (seconds > 0) View.VISIBLE else View.GONE
    alarmButton.visibility = if (this::mediaPlayer.isInitialized && mediaPlayer.isPlaying) View.VISIBLE else View.GONE
}

private fun runTimer() {
    val handler = Handler(Looper.getMainLooper())

    handler.post {
        if (running && seconds == 0) {
            playAlarm()
            refreshUIStates()
        } else if (running) {
            seconds--
            secondsView.setText("%02d".format(seconds % 60))
            minutesView.setText("%02d".format(seconds / 60))
        }

        handler.postDelayed({ runTimer() }, 1000)
    }
}
```

Kod 5 Obsługa timera w TImerFragment

```kotlin
private fun playAlarm() {
    if (!this::mediaPlayer.isInitialized) {
        mediaPlayer = MediaPlayer.create(requireContext(), R.raw.alarm_sound)
        mediaPlayer.isLooping = true
        initVolumeShaper()
    }
    audioManager = requireContext().getSystemService(AUDIO_SERVICE) as AudioManager
    audioManager.setStreamVolume(
        AudioManager.STREAM_MUSIC,
        audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC),
        0
    )

    volumeShaper.apply(VolumeShaper.Operation.PLAY)
    mediaPlayer.start()

    vibratePhone(300)
}

private fun initVolumeShaper() {
    val config: VolumeShaper.Configuration =
        VolumeShaper.Configuration.Builder()
            .setDuration(10000)
            .setCurve(floatArrayOf(0f, 1f), floatArrayOf(0f, 1f))
            .setInterpolatorType(VolumeShaper.Configuration.INTERPOLATOR_TYPE_CUBIC)
            .build()

    volumeShaper = mediaPlayer.createVolumeShaper(config)
}

private fun stopAlarm() {
    if (mediaPlayer.isPlaying) {
        mediaPlayer.pause()
        mediaPlayer.seekTo(0)
    }
}
```

Kod 6 Uruchamianie narastającego sygnału w TimerFragment

```kotlin
fun Fragment.vibratePhone(milliseconds: Long = 300) {
    val vib =
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            val vibratorManager =
                getSystemService(requireContext(), VibratorManager::class.java) as VibratorManager
            vibratorManager.defaultVibrator
        } else {
            @Suppress("DEPRECATION")
            getSystemService(requireContext(), Vibrator::class.java) as Vibrator
        }

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        vib.vibrate(VibrationEffect.createOneShot(milliseconds, VibrationEffect.DEFAULT_AMPLITUDE))
    } else {
        @Suppress("DEPRECATION") vib.vibrate(milliseconds)
    }
}
```

Kod 7 Uruchamianie wibracji w Vibrate.kt

Biblioteka wsparcia wzornictwa

Układ dla smartfonów na ekranie głównym składa się z paska aplikacji oraz kart (pomiędzy którymi można przechodzić za pomocą gestu przeciągania). Lista potraw używa RecyclerView z układem siatki. Potrawy zawierają zdjęcie i nazwę używając CardView.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    viewPager = view.findViewById(R.id.viewPager)
    viewPager!!.adapter = MyPagerAdapter(this)

    val tabLayout = view.findViewById<TabLayout>(R.id.tabs)
    TabLayoutMediator(tabLayout, viewPager!!) { tab, position ->
        when (position) {
            0 -> tab.text = "Home"
            1 -> tab.text = RecipeCategory.Breakfast.toString()
            2 -> tab.text = RecipeCategory.Dinner.toString()
        }
    }.attach()
}

class MyPagerAdapter(fragment: Fragment) : FragmentStateAdapter(fragment) {
    override fun getItemCount(): Int = 3

    companion object {
        var selectedTabPosition: Int? = null
    }

    override fun createFragment(position: Int): Fragment {
        return when (position) {
            0 -> HomeFragment.newInstance()
            1 -> RecipeRecycleViewFragment.newInstance(RecipeCategory.Breakfast)
            2 -> RecipeRecycleViewFragment.newInstance(RecipeCategory.Dinner)
            else -> HomeFragment.newInstance()
        }
    }
}
```

Kod 8 Tworzenie kart z gestem przeciągania (ViewPager2) w RecipeTypesTabsFragment

```
<com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

    <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            app:title="Recipes" />

    <com.google.android.material.tabs.TabLayout
            android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:tabMode="fixed"
            app:tabGravity="fill" />

</com.google.android.material.appbar.AppBarLayout>

<androidx.viewpager2.widget.ViewPager2
        android:id="@+id/viewPager"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

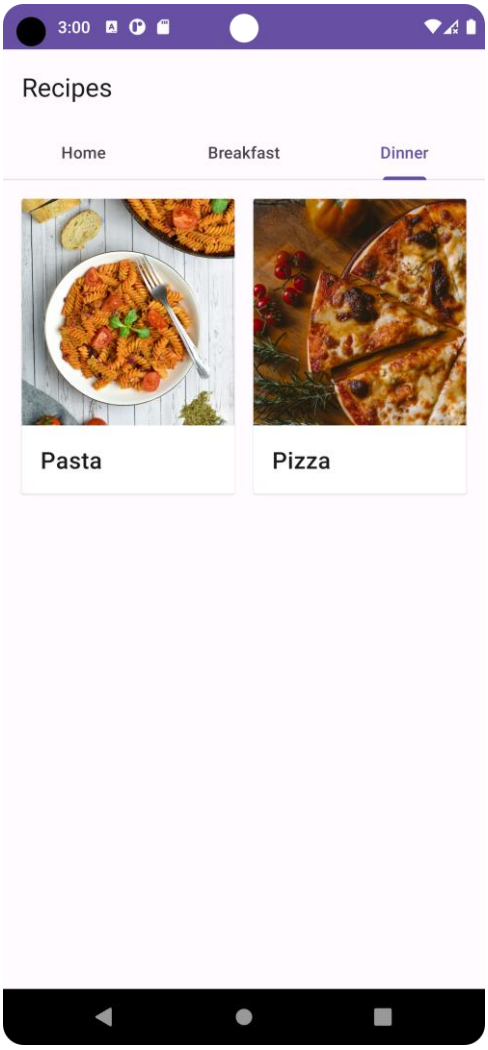
Kod 9 XML RecipeTypesTabsFragment

```
class MyViewHolder(view: View, private val listener: RecipeListFragment.Companion.Listener?) :
    RecyclerView.ViewHolder(view) {
    private val imageView = view.findViewById<ImageView>(R.id.imageView)
    private val nameTextView = view.findViewById<TextView>(R.id.nameTextView)
    private val cardView = view.findViewById<CardView>(R.id.cardView)

    fun bind(item: Recipe) {
        imageView.setImageResource(item.imageId)
        nameTextView.text = item.name
        cardView.setOnClickListener {
            this.listener?.itemClicked(item.uid)
        }
    }
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    val category = this.category ?: return
    val recyclerView = view.findViewById<RecyclerView>(R.id.recyclerView)
    recyclerView.layoutManager = GridLayoutManager(context, 2)
    recyclerView.adapter = MyAdapter(emptyList(), this.listener)

    AppDatabase.getDatabase(requireContext())
        .recipeDao()
        .getAll(category)
        .observe(viewLifecycleOwner) { items ->
            recyclerView.adapter = MyAdapter(items, this.listener)
        }
}

class MyAdapter(private val items: List<Recipe>, private val listener: RecipeListFragment.Companion.Listener?) :
    RecyclerView.Adapter<MyViewHolder>() {
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_recipe, parent, false)
        return MyViewHolder(view, this.listener)
    }

    override fun getItemCount(): Int = items.size
    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        val item = items[position]
        holder.bind(item)
    }
}
```

Kod 10 Tworzenie RecyclerView w RecipeRecycleViewFragment

```xml
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
                                   android:layout_width="match_parent"
                                   android:layout_height="wrap_content"
                                   xmlns:app="http://schemas.android.com/apk/res-auto"
                                   android:layout_margin="8dp"
                                   app:cardElevation="1dp"
                                   android:id="@+id/cardView"
                                   android:clickable="true"
                                   android:foreground="?android:attr/selectableItemBackground"
        >

    <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

        <ImageView
                android:id="@+id/imageView"
                android:layout_width="match_parent"
                android:layout_height="194dp"
                android:scaleType="centerCrop"
                android:src="@drawable/baseline_stop_24"
                android:contentDescription="@string/recipe_photo"/>

        <TextView
                android:id="@+id/nameTextView"
                android:padding="16dp"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/recipe_name"
                android:textAppearance="?attr/textAppearanceHeadline6"
                />

    </LinearLayout>

</androidx.cardview.widget.CardView>
```

Kod 11 XML dla CardView (item_recipe)



Zdjęcie 9 Widok siatki

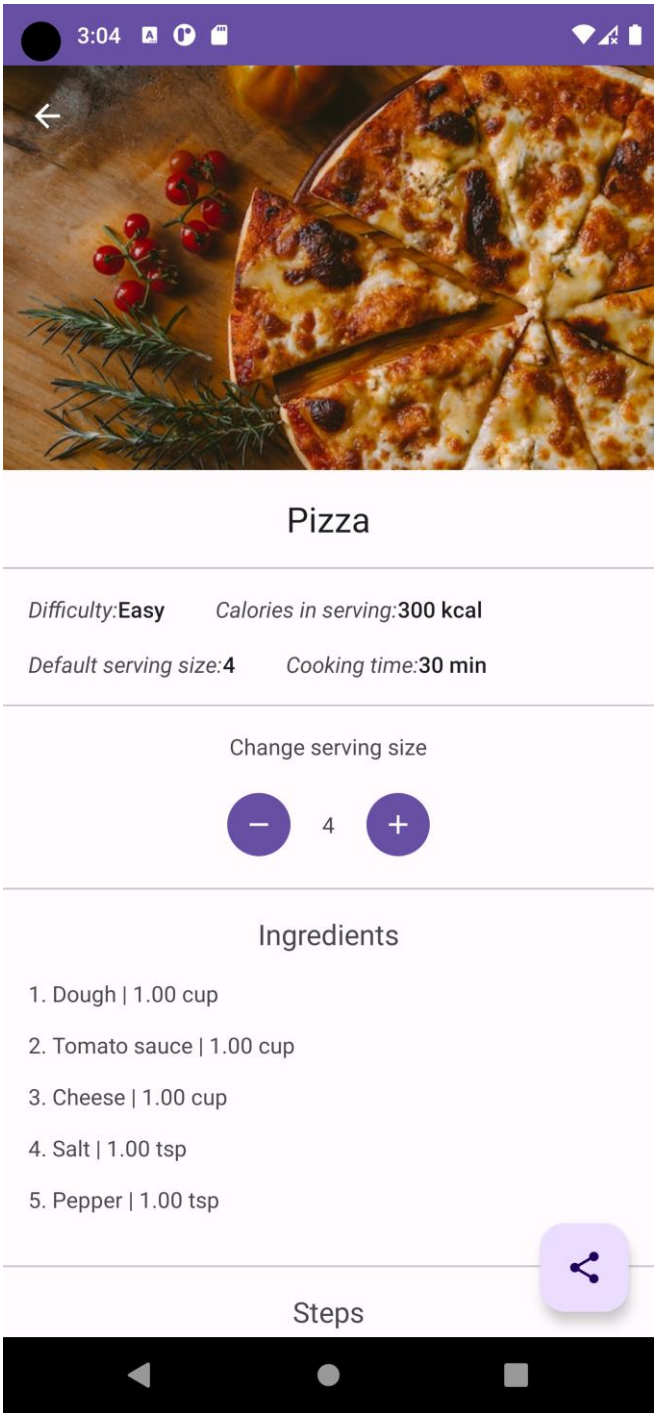
Widok szczegółów potrawy zawiera pasek aplikacji, w którym jest zdjęcie oraz FAB. Zdjęcie się zwija przy przewijaniu ekranu.

```
<com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="256dp"
        app:liftOnScroll="true">

    <com.google.android.material.appbar.CollapsingToolbarLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            app:contentScrim="?attr/colorPrimary"
            app:statusBarScrim="?attr/colorPrimaryVariant"
            app:layout_scrollFlags="scroll|exitUntilCollapsed|snap"
            app:expandedTitleTextColor="@android:color/transparent"
            app:collapsedTitleTextColor="@color/design_default_color_on_primary">

        <ImageView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:scaleType="centerCrop"
                android:src="@drawable/pizza"
                android:id="@+id/toolbar_recipe_image"
                android:fitsSystemWindows="true"
                android:contentDescription="@string/recipe_image"
                />

        <com.google.android.material.appbar.MaterialToolbar
                android:id="@+id/topAppBar"
                app:layout_collapseMode="pin"
                android:background="@android:color/transparent"
                android:elevation="1dp"
                app:layout_scrollFlags="scroll|enterAlways|snap"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                app:navigationIconTint="@color/design_default_color_on_primary"
                app:navigationIcon="@drawable/baseline_arrow_back_24"
                />


    </com.google.android.material.appbar.CollapsingToolbarLayout>


</com.google.android.material.appbar.AppBarLayout>
```

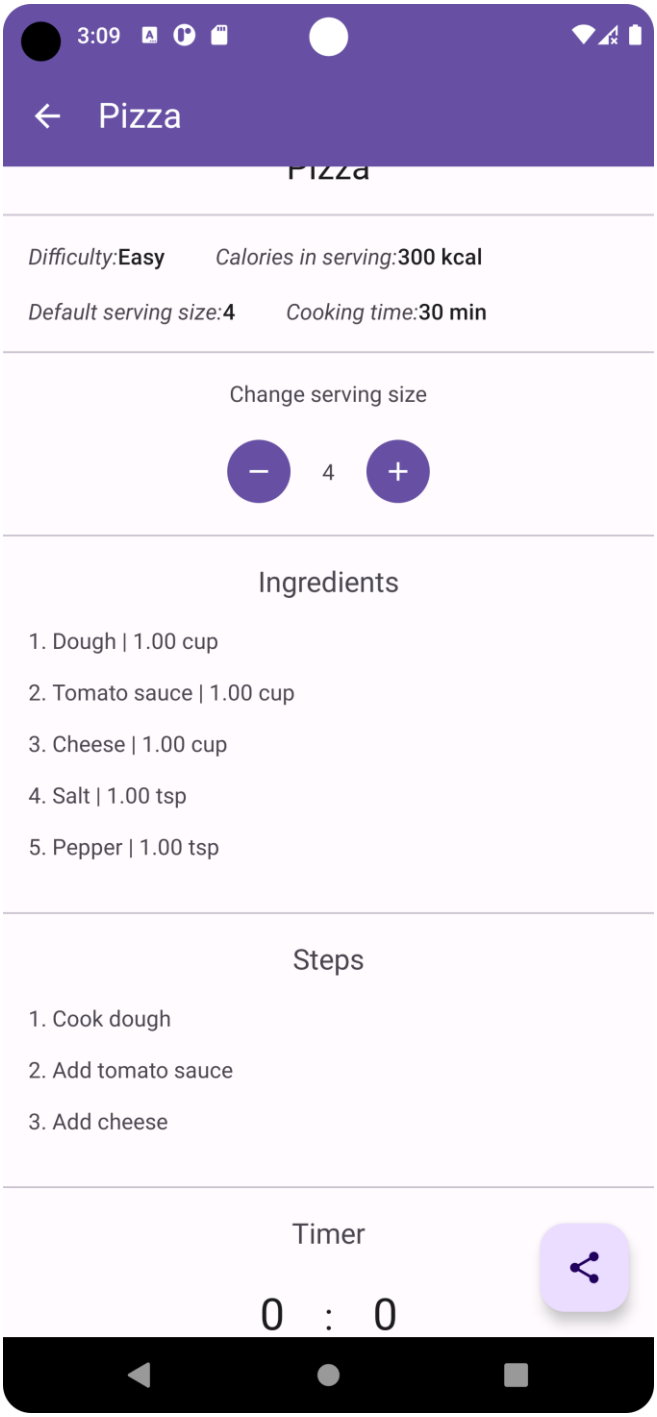
Kod 12 Pasek aplikacji ze zwijanym zdjęciem w fragment_recipe_detail

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/fab"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        android:src="@drawable/baseline_share_24"
        android:contentDescription="@string/fab_content_desc"/>
```

Kod 13 FAB w fragment_recipe_detail



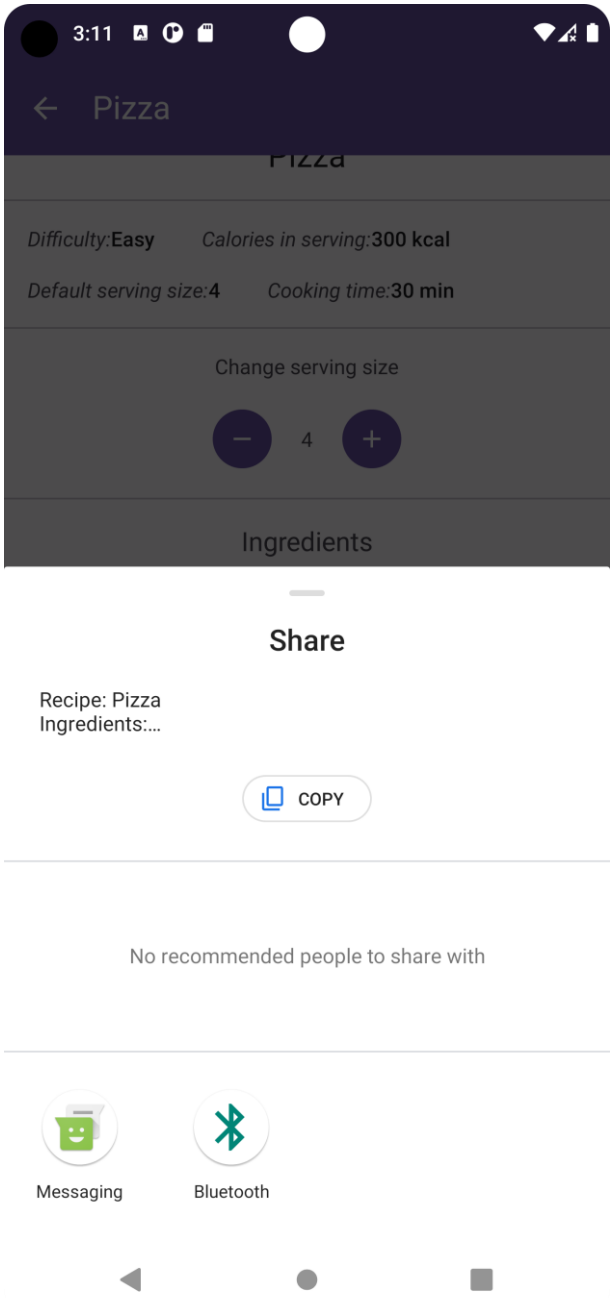
Zdjęcie 10 Widok szczegółów potrawy z obrazkiem na pasku oraz FAB



Zdjęcie 11 Widok szczegółów potrawy ze zwiniętym zdjęciem

```kotlin
private fun onFabClick(view: View) {
    val sendIntent: Intent = Intent().apply {
        action = Intent.ACTION_SEND
        putExtra(Intent.EXTRA_TEXT, recipeIngredientsShare)
        type = "text/plain"
    }

    val shareIntent = Intent.createChooser(sendIntent, "Share recipe")
    startActivity(shareIntent)
}
```

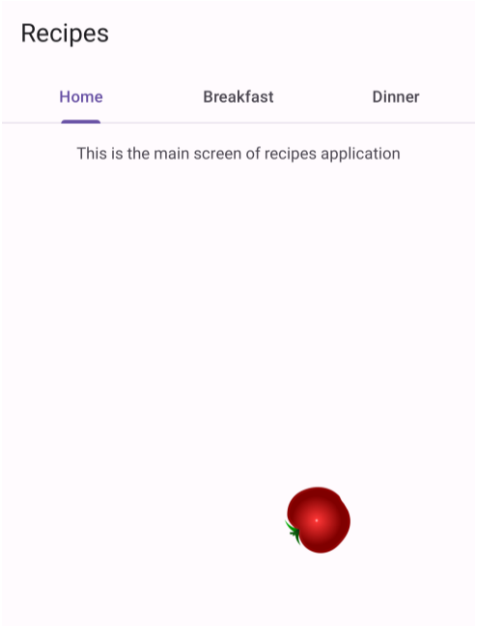
Kod 14 Wyświetlanie ekranu udostępniania przy klikaniu FAB



Zdjęcie 12 Widok ekranu udostępniania po kliknięciu FAB

Animacje

Na ekranie startowym układu dla smartfonów znajduje się prosta animacja przedstawiająca toczącego się pomidora.



Zdjęcie 13 Toczący się pomidor na stronie startowej

```kotlin
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    val view = inflater.inflate(R.layout.fragment_home, container, false)

    imageView = view.findViewById(R.id.tomatoe_img)

    val animationMove = ObjectAnimator.ofFloat(imageView, "translationX", -400f, 400f).apply {
        duration = 4000
        repeatCount = ObjectAnimator.INFINITE
        repeatMode = ObjectAnimator.REVERSE
    }

    val animationRotate = ObjectAnimator.ofFloat(imageView, "rotation", -400f, 400f).apply {
        duration = 4000
        repeatCount = ObjectAnimator.INFINITE
        repeatMode = ObjectAnimator.REVERSE
    }

    val set = AnimatorSet()
    set.playTogether(animationMove, animationRotate)
    set.start()

    return view;
}
```

Kod 15 Implementacja animacji w HomeFragment@onCreateView

Obsługa zmiany orientacji urządzenia

W większości fragmentów / aktywności można znaleźć kod obsługujący zmianę konfiguracji urządzenia (w tym orientację). Kod często jest bardzo do siebie podobny, więc na załączonych zdjęciach został pokazany fragment dla minutnika.

```
1  private var seconds = 0
2  private var running = false
3  private var wasRunning = false
4
5  override fun onCreate(savedInstanceState: Bundle?) {
6      super.onCreate(savedInstanceState)
7
8      savedInstanceState?.let {
9          seconds = it.getInt(ARG_PARAM_SECONDS)
10         running = it.getBoolean(ARG_PARAM_RUNNING)
11         wasRunning = it.getBoolean(ARG_PARAM_WAS_RUNNING)
12     }
13 }
14
15 override fun onSaveInstanceState(outState: Bundle) {
16   super.onSaveInstanceState(outState)
17   outState.putInt(ARG_PARAM_SECONDS, seconds)
18   outState.putBoolean(ARG_PARAM_RUNNING, running)
19   outState.putBoolean(ARG_PARAM_WAS_RUNNING, wasRunning)
20 }
```

Kod 16 Kod obsługi zmiany orientacji urządzenia w TimerFragment