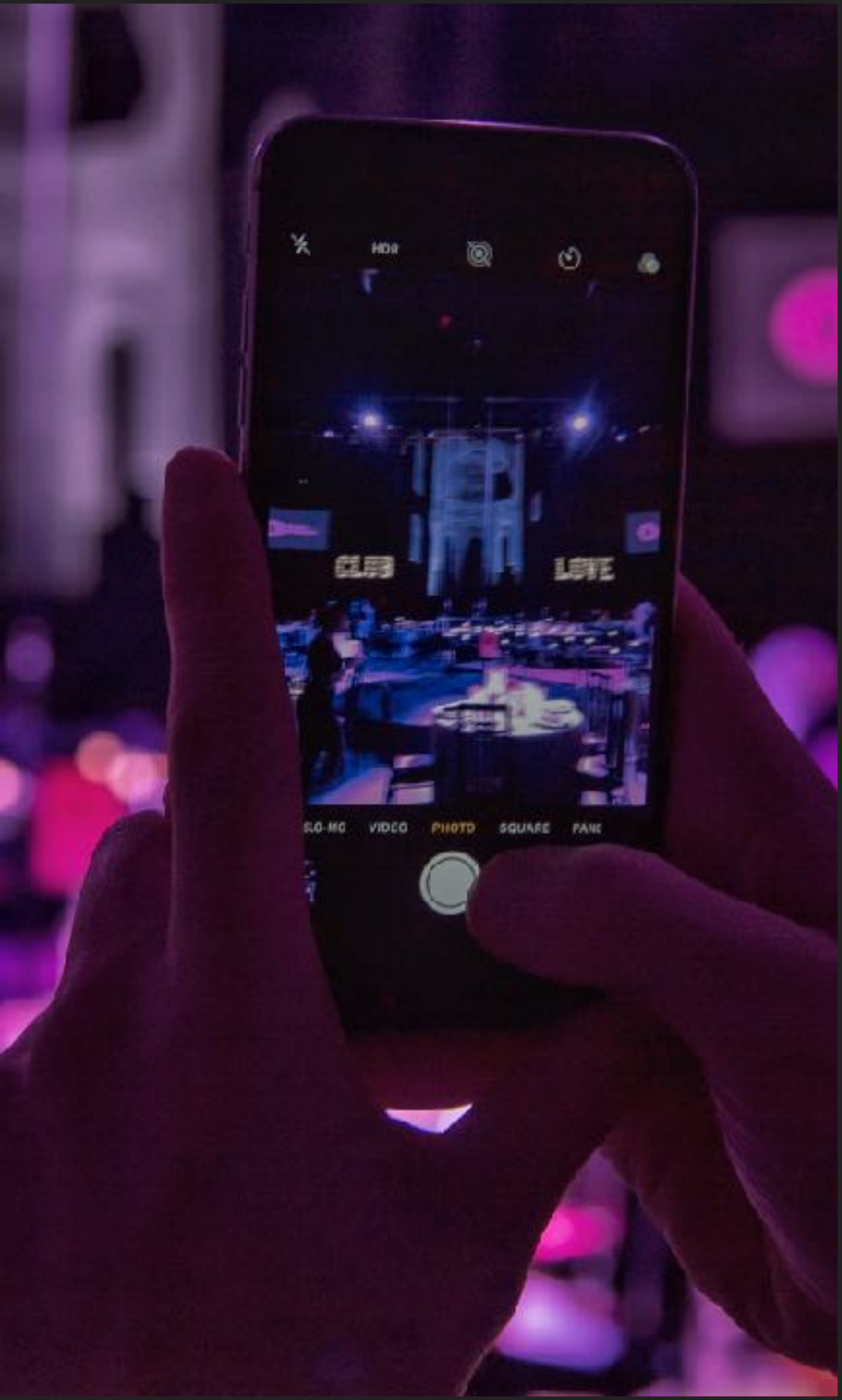
Android Developers Italia

ALEX GOTEV

DECLARATIVE DYNAMIC UI

01/04/2020



github.com/gotev

linkedin.com/in/alexgotev

@alexgt89

ALEX GOTEV

Senior Mobile Engineer @ Sky Italia

Not so long ago in an house nearby. . . .

github.com/gotev/swapi-android

ENTER

SWAPI.CO

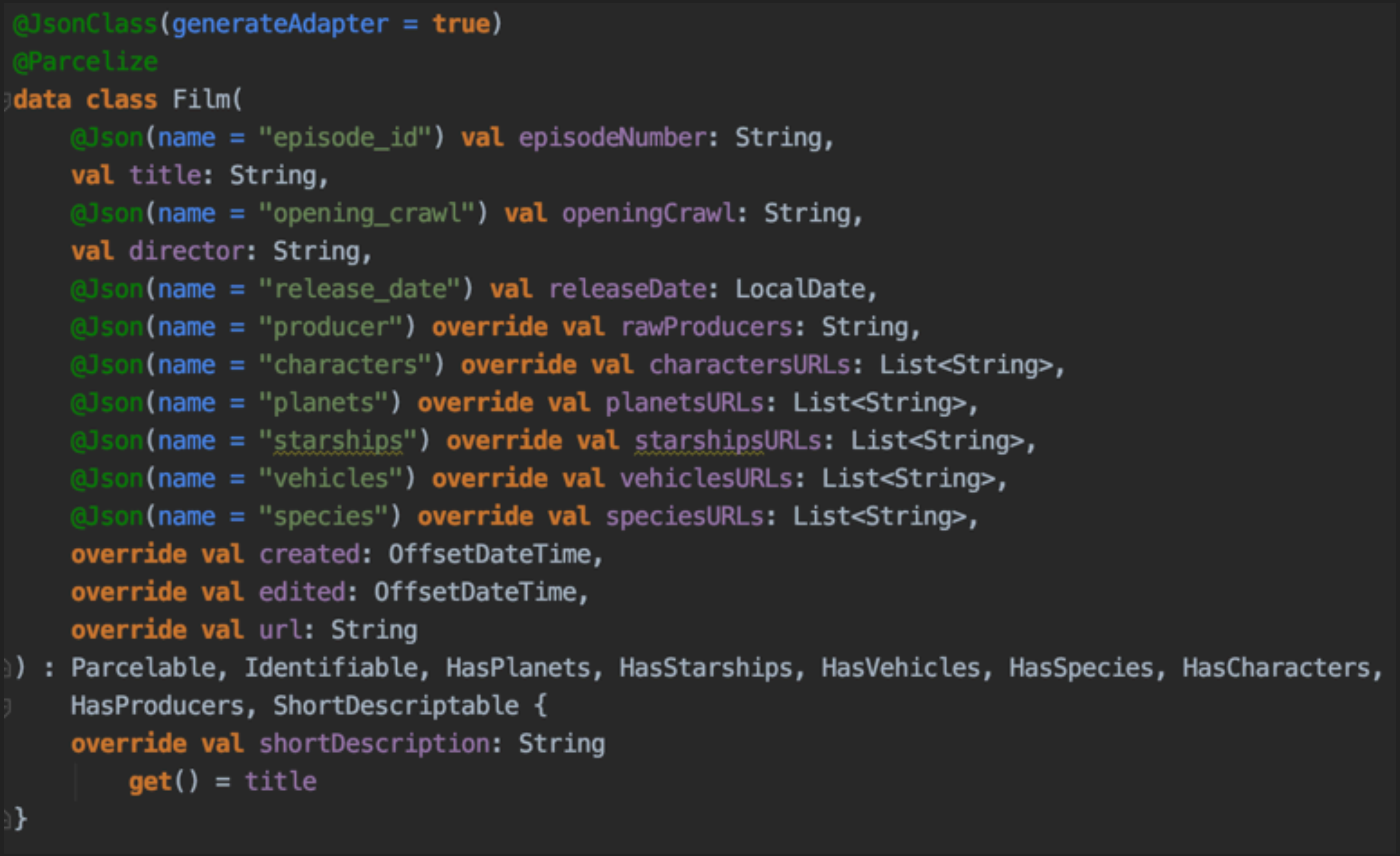
TECHNOLOGY STACK

- Kotlin + Coroutines
- OkHttp/Moshi/Retrofit/ThreeTenABP
- JetPack
    - AppCompat
    - Lifecycle + LiveData & ViewModels
    - Navigation
- Material
- Recycler Adapter

DOMAIN MODELS

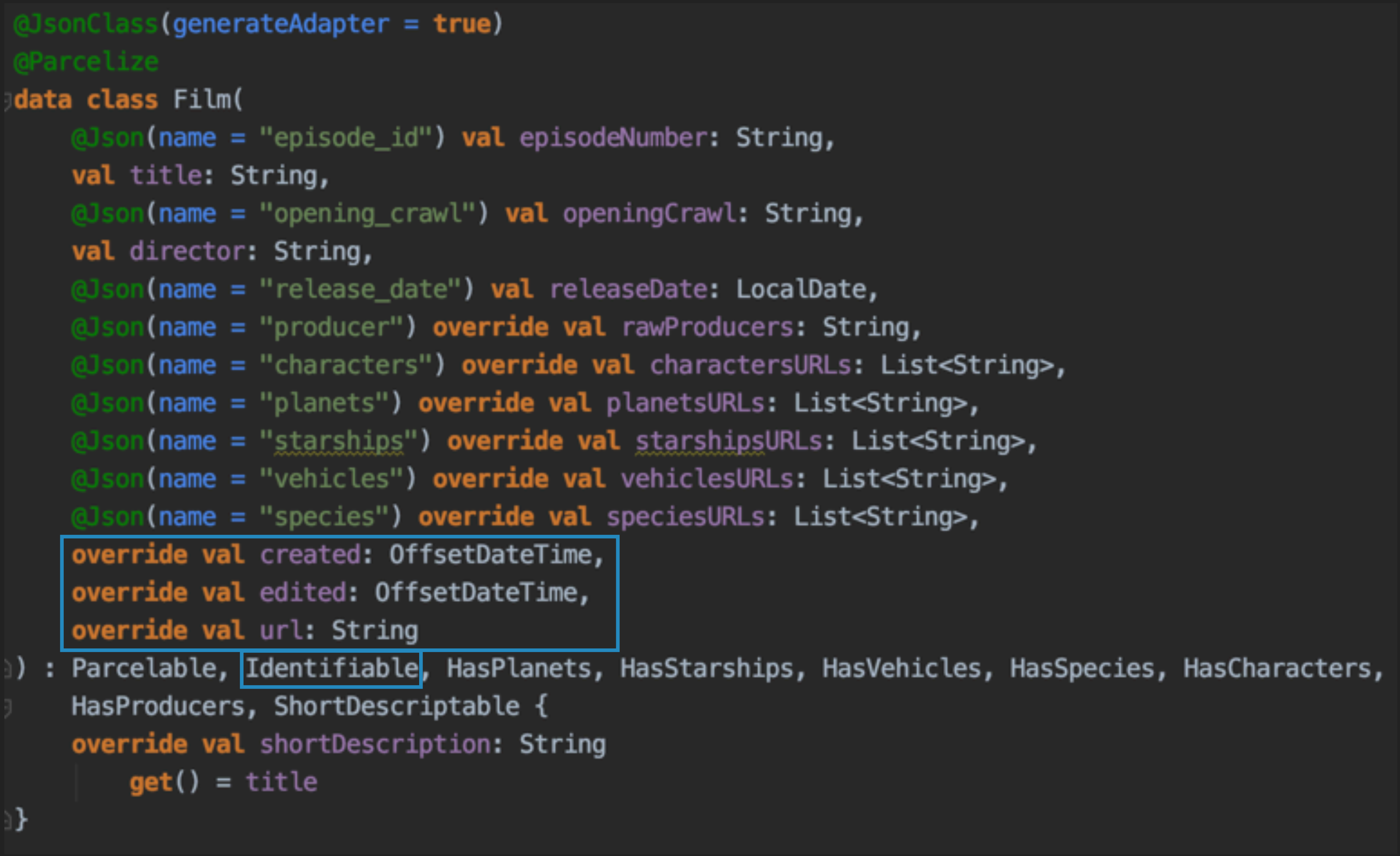
- Films
- Characters
- Species
- Planets
- Vehicles
- Starships

FILM MODEL



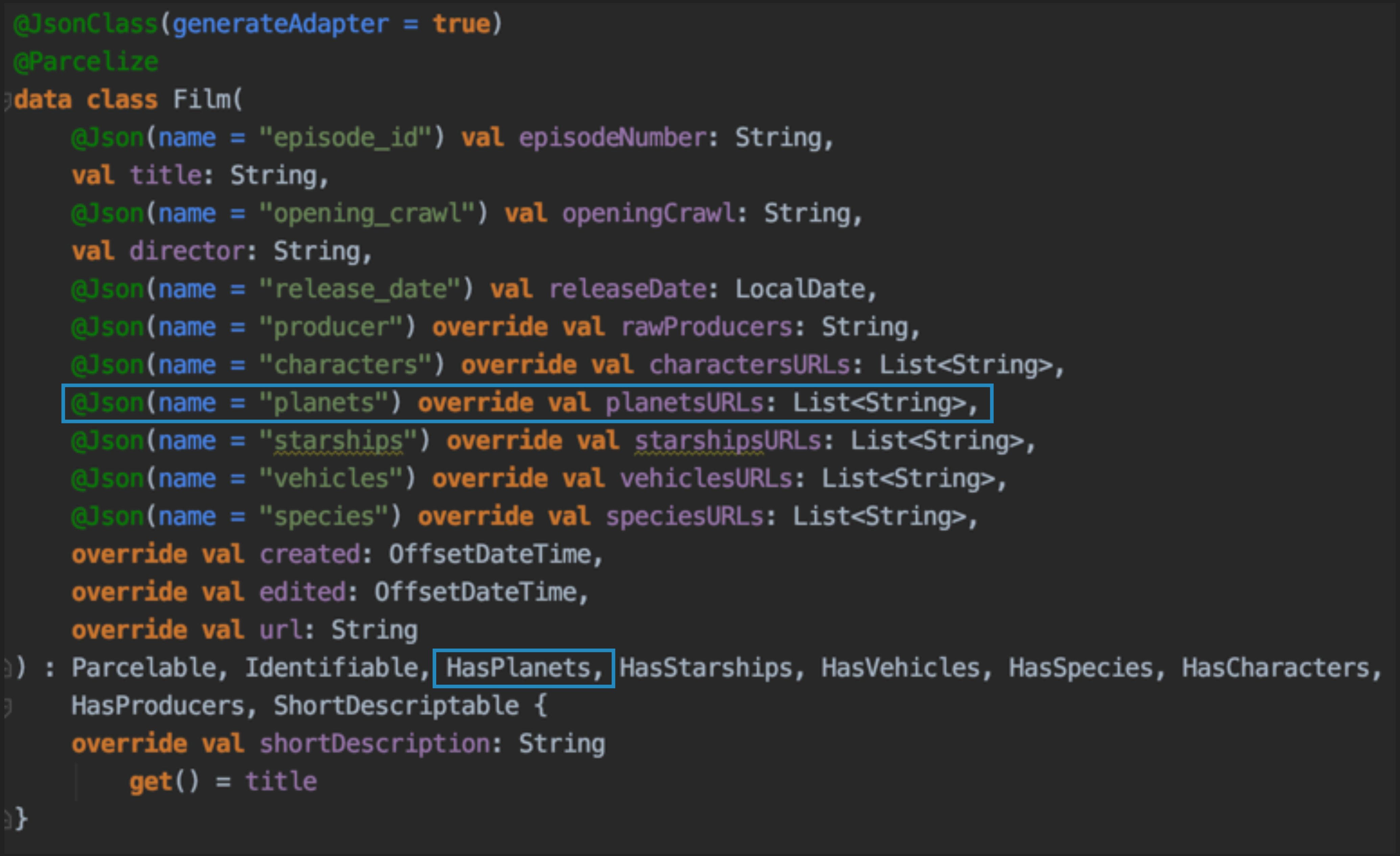
```kotlin
@JsonClass(generateAdapter = true)
@Parcelize
data class Film(
    @Json(name = "episode_id") val episodeNumber: String,
    val title: String,
    @Json(name = "opening_crawl") val openingCrawl: String,
    val director: String,
    @Json(name = "release_date") val releaseDate: LocalDate,
    @Json(name = "producer") override val rawProducers: String,
    @Json(name = "characters") override val charactersURLs: List<String>,
    @Json(name = "planets") override val planetsURLs: List<String>,
    @Json(name = "starships") override val starshipsURLs: List<String>,
    @Json(name = "vehicles") override val vehiclesURLs: List<String>,
    @Json(name = "species") override val speciesURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasPlanets, HasStarships, HasVehicles, HasSpecies, HasCharacters,
    HasProducers, ShortDescriptable {
    override val shortDescription: String
        get() = title
}
```

FILM MODEL



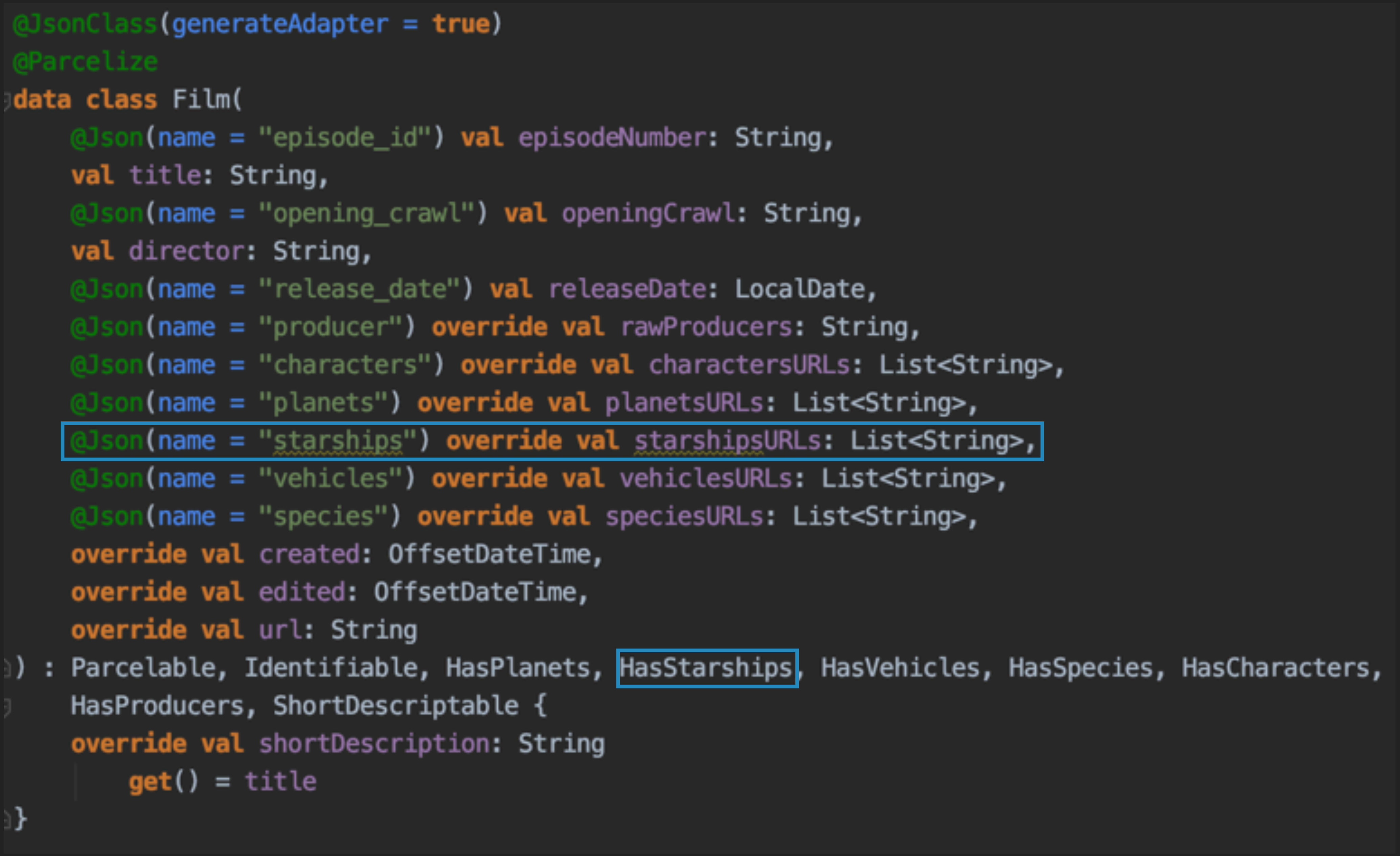
```kotlin
@JsonClass(generateAdapter = true)
@Parcelize
data class Film(
    @Json(name = "episode_id") val episodeNumber: String,
    val title: String,
    @Json(name = "opening_crawl") val openingCrawl: String,
    val director: String,
    @Json(name = "release_date") val releaseDate: LocalDate,
    @Json(name = "producer") override val rawProducers: String,
    @Json(name = "characters") override val charactersURLs: List<String>,
    @Json(name = "planets") override val planetsURLs: List<String>,
    @Json(name = "starships") override val starshipsURLs: List<String>,
    @Json(name = "vehicles") override val vehiclesURLs: List<String>,
    @Json(name = "species") override val speciesURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasPlanets, HasStarships, HasVehicles, HasSpecies, HasCharacters,
    HasProducers, ShortDescriptable {
    override val shortDescription: String
        get() = title
}
```

FILM MODEL



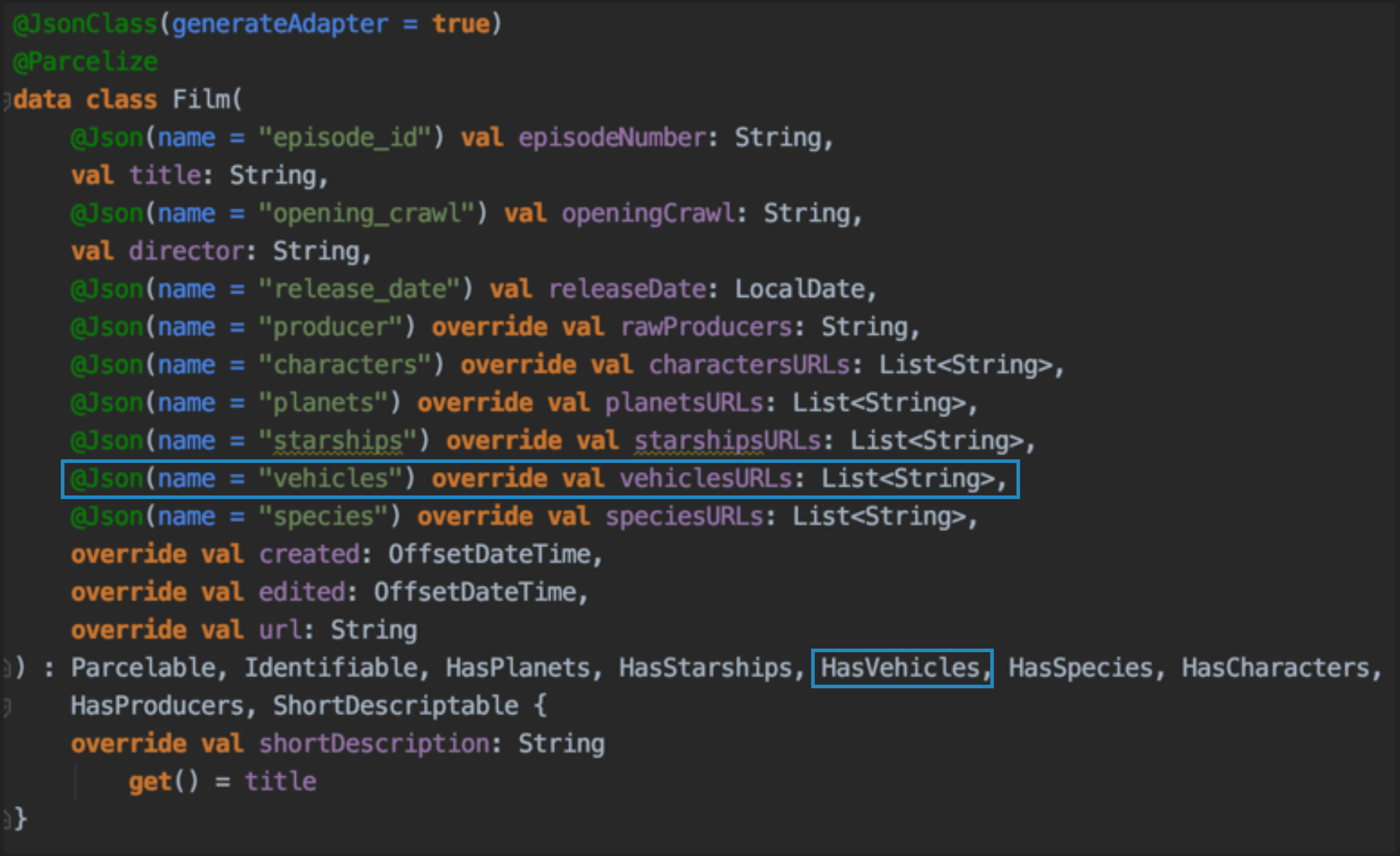
```
@JsonClass(generateAdapter = true)
@Parcelize
data class Film(
    @Json(name = "episode_id") val episodeNumber: String,
    val title: String,
    @Json(name = "opening_crawl") val openingCrawl: String,
    val director: String,
    @Json(name = "release_date") val releaseDate: LocalDate,
    @Json(name = "producer") override val rawProducers: String,
    @Json(name = "characters") override val charactersURLs: List<String>,
    @Json(name = "planets") override val planetsURLs: List<String>,
    @Json(name = "starships") override val starshipsURLs: List<String>,
    @Json(name = "vehicles") override val vehiclesURLs: List<String>,
    @Json(name = "species") override val speciesURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasPlanets, HasStarships, HasVehicles, HasSpecies, HasCharacters,
    HasProducers, ShortDescriptable {
    override val shortDescription: String
        get() = title
}
```

FILM MODEL



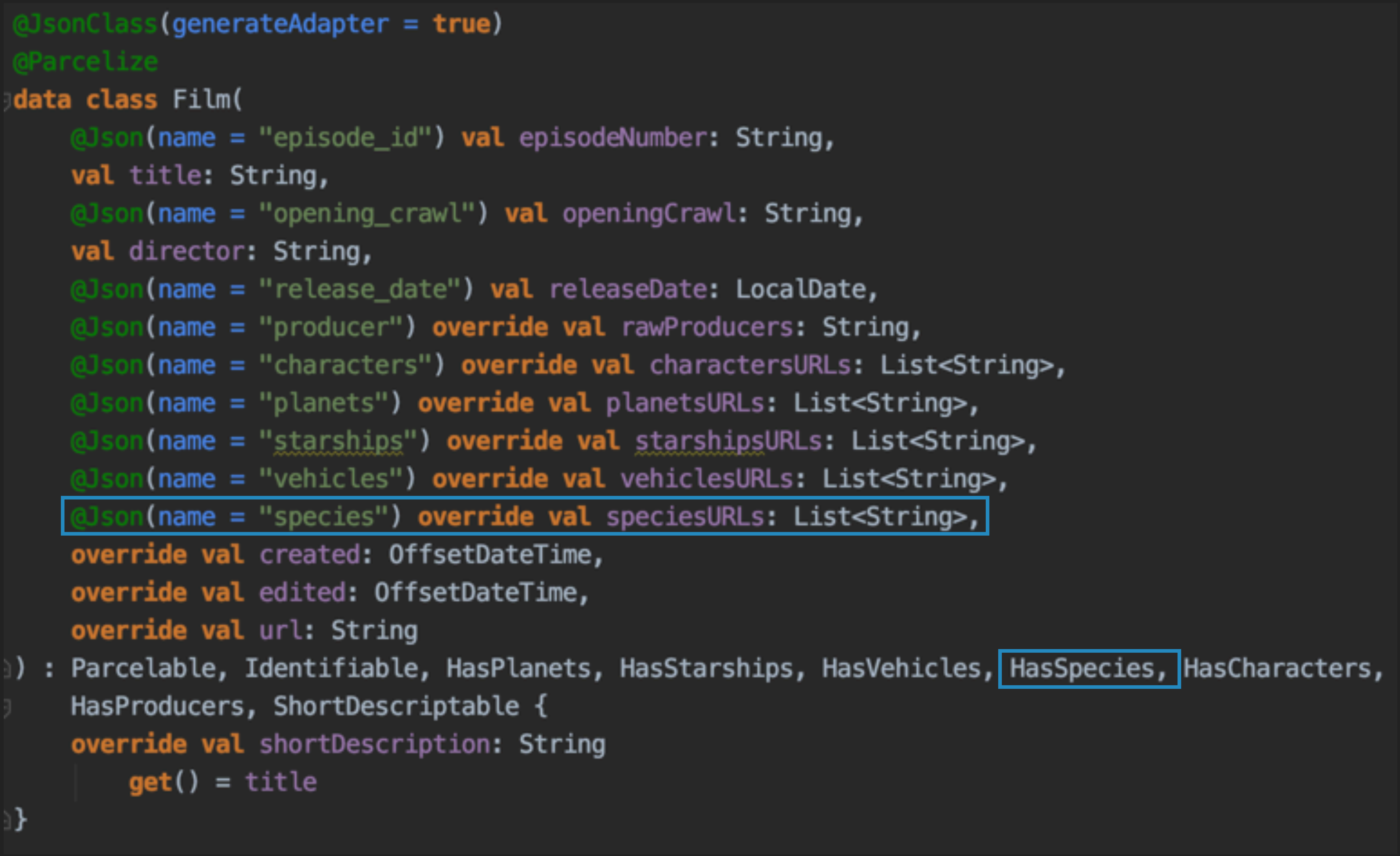
```kotlin
@JsonClass(generateAdapter = true)
@Parcelize
data class Film(
    @Json(name = "episode_id") val episodeNumber: String,
    val title: String,
    @Json(name = "opening_crawl") val openingCrawl: String,
    val director: String,
    @Json(name = "release_date") val releaseDate: LocalDate,
    @Json(name = "producer") override val rawProducers: String,
    @Json(name = "characters") override val charactersURLs: List<String>,
    @Json(name = "planets") override val planetsURLs: List<String>,
    @Json(name = "starships") override val starshipsURLs: List<String>,
    @Json(name = "vehicles") override val vehiclesURLs: List<String>,
    @Json(name = "species") override val speciesURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasPlanets, HasStarships, HasVehicles, HasSpecies, HasCharacters,
    HasProducers, ShortDescriptable {
    override val shortDescription: String
        get() = title
}
```

FILM MODEL



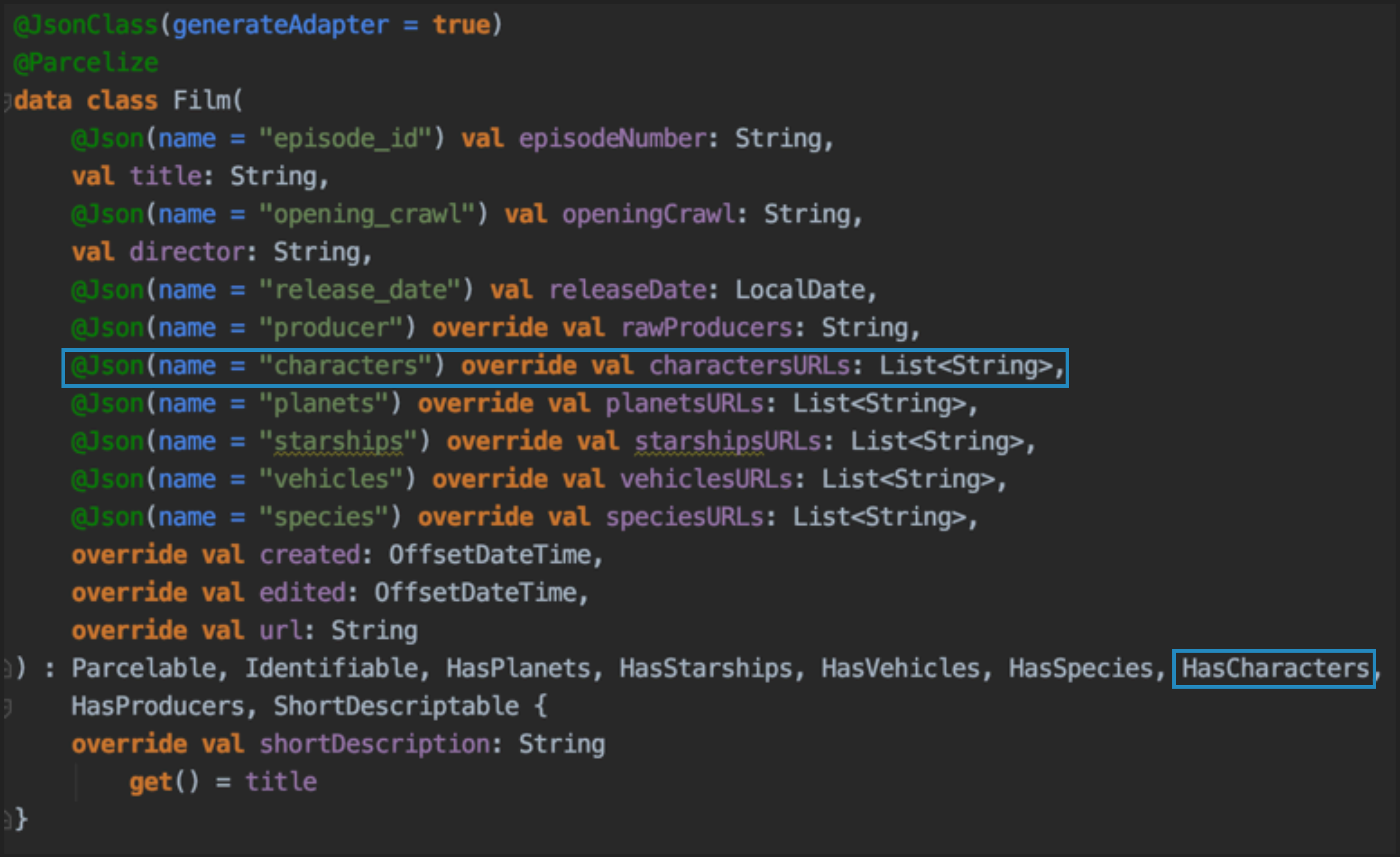
```
@JsonClass(generateAdapter = true)
@Parcelize
data class Film(
    @Json(name = "episode_id") val episodeNumber: String,
    val title: String,
    @Json(name = "opening_crawl") val openingCrawl: String,
    val director: String,
    @Json(name = "release_date") val releaseDate: LocalDate,
    @Json(name = "producer") override val rawProducers: String,
    @Json(name = "characters") override val charactersURLs: List<String>,
    @Json(name = "planets") override val planetsURLs: List<String>,
    @Json(name = "starships") override val starshipsURLs: List<String>,
    @Json(name = "vehicles") override val vehiclesURLs: List<String>,
    @Json(name = "species") override val speciesURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasPlanets, HasStarships, HasVehicles, HasSpecies, HasCharacters,
    HasProducers, ShortDescriptable {
    override val shortDescription: String
        get() = title
}
```

FILM MODEL



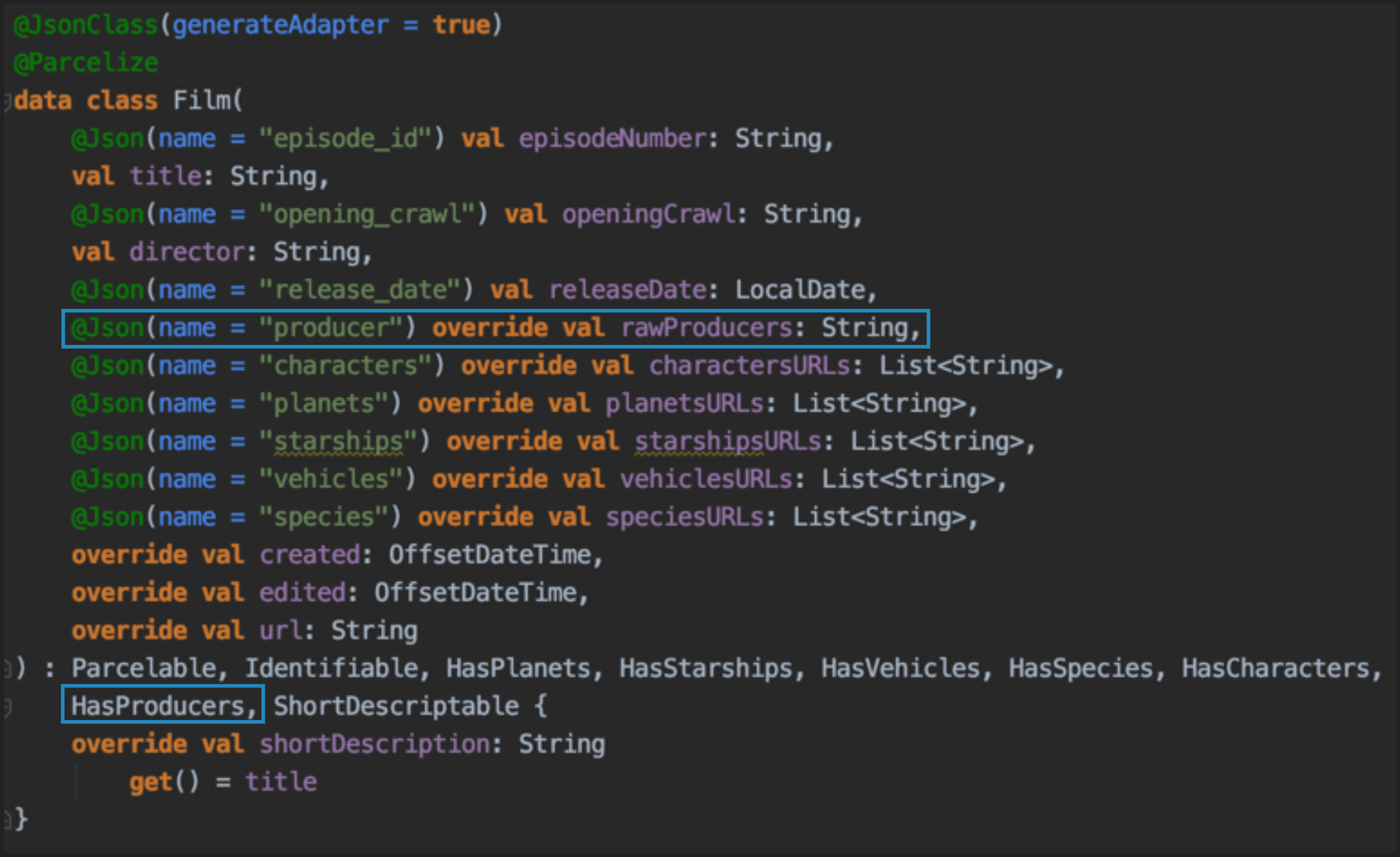
```kotlin
@JsonClass(generateAdapter = true)
@Parcelize
data class Film(
    @Json(name = "episode_id") val episodeNumber: String,
    val title: String,
    @Json(name = "opening_crawl") val openingCrawl: String,
    val director: String,
    @Json(name = "release_date") val releaseDate: LocalDate,
    @Json(name = "producer") override val rawProducers: String,
    @Json(name = "characters") override val charactersURLs: List<String>,
    @Json(name = "planets") override val planetsURLs: List<String>,
    @Json(name = "starships") override val starshipsURLs: List<String>,
    @Json(name = "vehicles") override val vehiclesURLs: List<String>,
    @Json(name = "species") override val speciesURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasPlanets, HasStarships, HasVehicles, HasSpecies, HasCharacters,
    HasProducers, ShortDescriptable {
    override val shortDescription: String
        get() = title
}
```

FILM MODEL



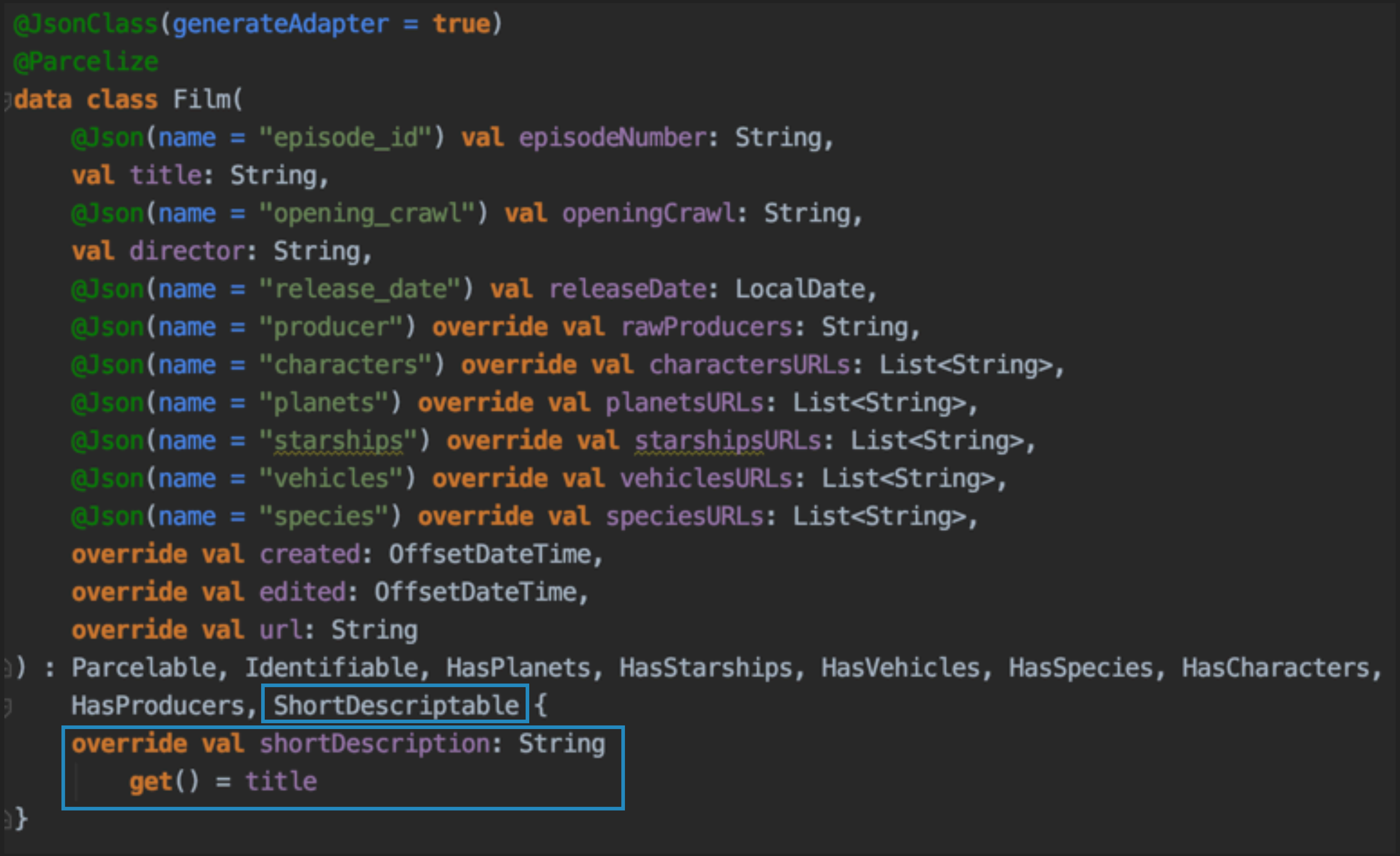
```kotlin
@JsonClass(generateAdapter = true)
@Parcelize
data class Film(
    @Json(name = "episode_id") val episodeNumber: String,
    val title: String,
    @Json(name = "opening_crawl") val openingCrawl: String,
    val director: String,
    @Json(name = "release_date") val releaseDate: LocalDate,
    @Json(name = "producer") override val rawProducers: String,
    @Json(name = "characters") override val charactersURLs: List<String>,
    @Json(name = "planets") override val planetsURLs: List<String>,
    @Json(name = "starships") override val starshipsURLs: List<String>,
    @Json(name = "vehicles") override val vehiclesURLs: List<String>,
    @Json(name = "species") override val speciesURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasPlanets, HasStarships, HasVehicles, HasSpecies, HasCharacters,
    HasProducers, ShortDescriptable {
    override val shortDescription: String
        get() = title
}
```

FILM MODEL



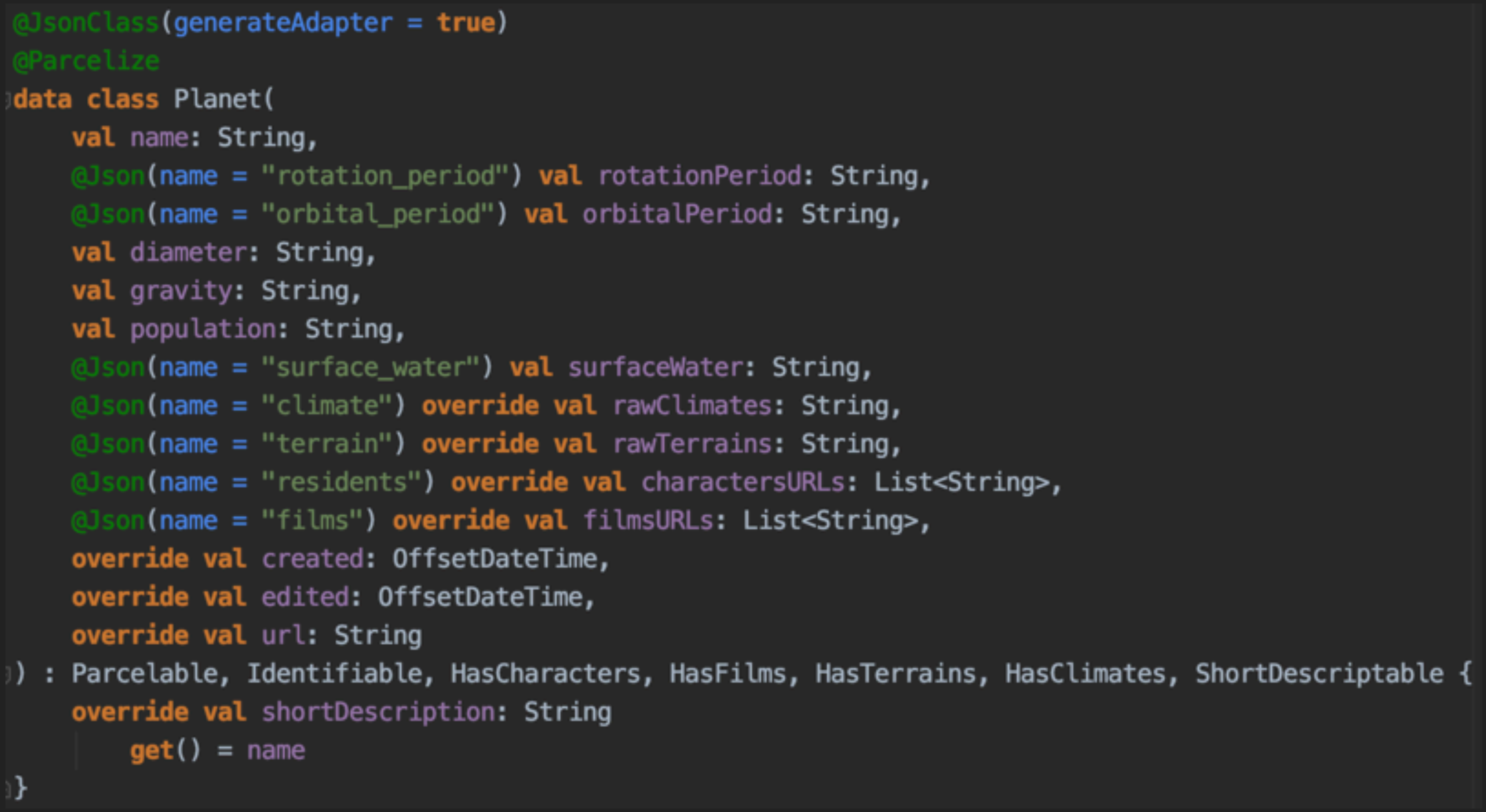
```kotlin
@JsonClass(generateAdapter = true)
@Parcelize
data class Film(
    @Json(name = "episode_id") val episodeNumber: String,
    val title: String,
    @Json(name = "opening_crawl") val openingCrawl: String,
    val director: String,
    @Json(name = "release_date") val releaseDate: LocalDate,
    @Json(name = "producer") override val rawProducers: String,
    @Json(name = "characters") override val charactersURLs: List<String>,
    @Json(name = "planets") override val planetsURLs: List<String>,
    @Json(name = "starships") override val starshipsURLs: List<String>,
    @Json(name = "vehicles") override val vehiclesURLs: List<String>,
    @Json(name = "species") override val speciesURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasPlanets, HasStarships, HasVehicles, HasSpecies, HasCharacters,
    HasProducers, ShortDescriptable {
    override val shortDescription: String
        get() = title
}
```

FILM MODEL



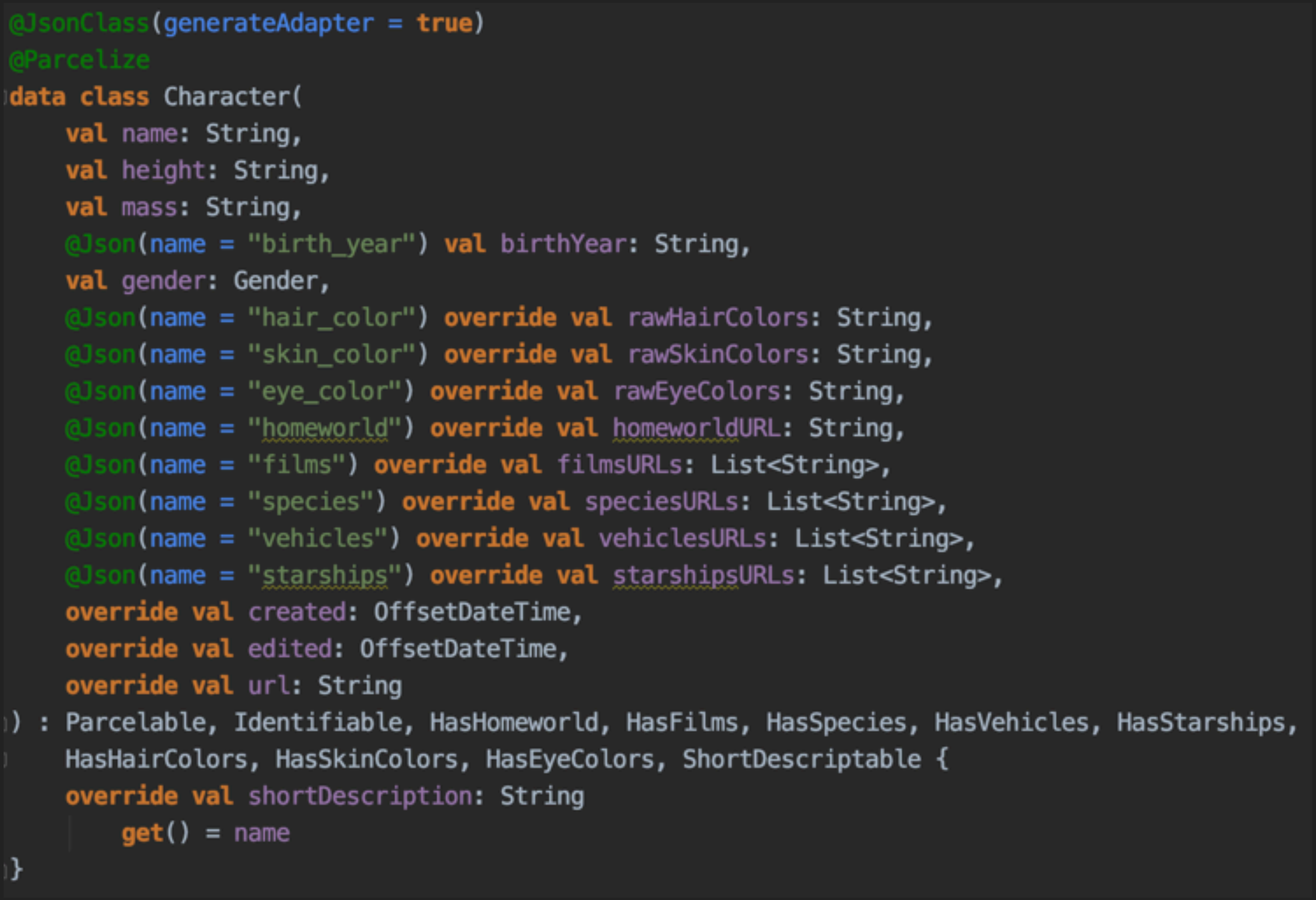
```kotlin
@JsonClass(generateAdapter = true)
@Parcelize
data class Film(
    @Json(name = "episode_id") val episodeNumber: String,
    val title: String,
    @Json(name = "opening_crawl") val openingCrawl: String,
    val director: String,
    @Json(name = "release_date") val releaseDate: LocalDate,
    @Json(name = "producer") override val rawProducers: String,
    @Json(name = "characters") override val charactersURLs: List<String>,
    @Json(name = "planets") override val planetsURLs: List<String>,
    @Json(name = "starships") override val starshipsURLs: List<String>,
    @Json(name = "vehicles") override val vehiclesURLs: List<String>,
    @Json(name = "species") override val speciesURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasPlanets, HasStarships, HasVehicles, HasSpecies, HasCharacters,
    HasProducers, ShortDescriptable {
    override val shortDescription: String
        get() = title
}
```

PLANET MODEL



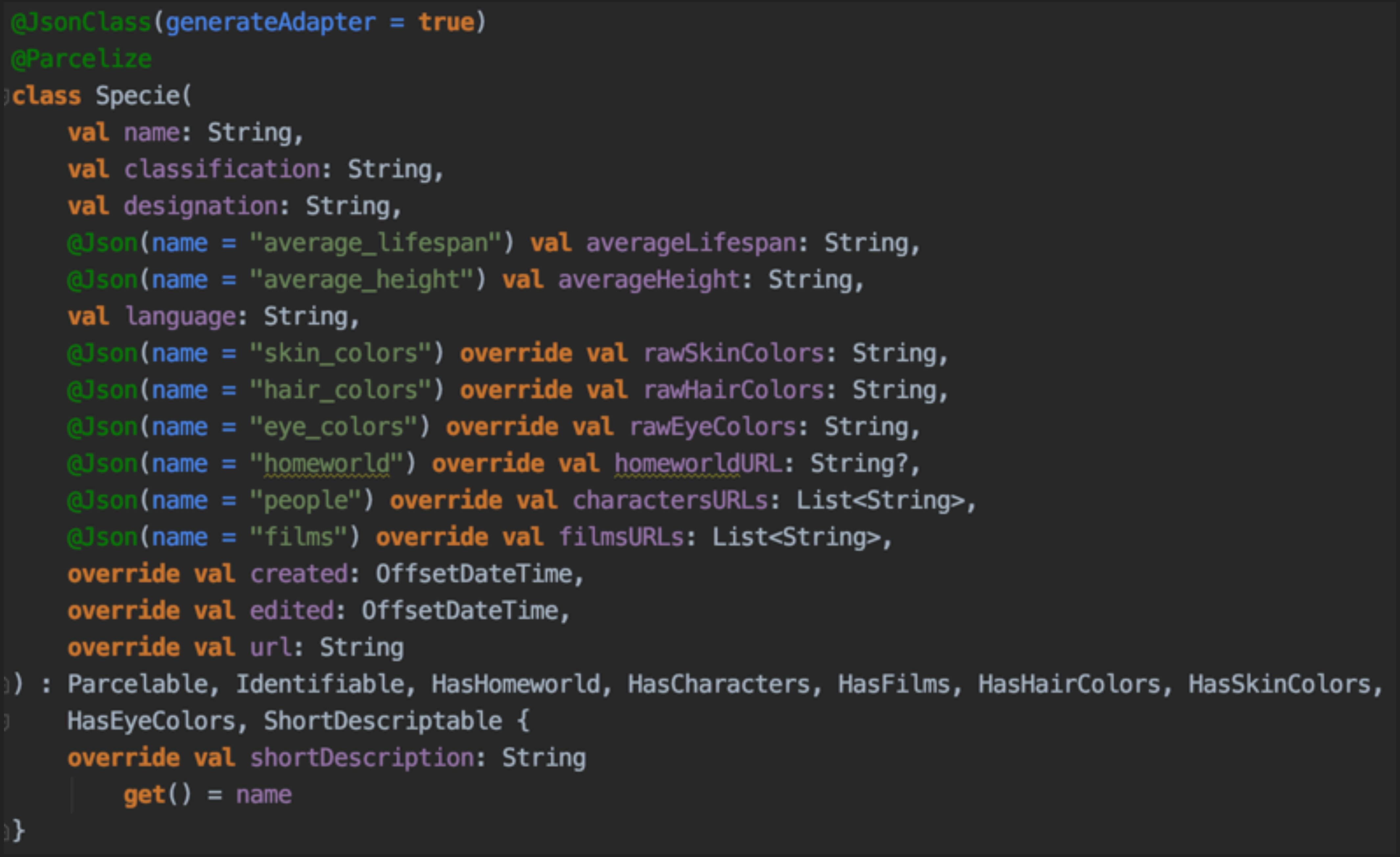
```kotlin
@JsonClass(generateAdapter = true)
@Parcelize
data class Planet(
    val name: String,
    @Json(name = "rotation_period") val rotationPeriod: String,
    @Json(name = "orbital_period") val orbitalPeriod: String,
    val diameter: String,
    val gravity: String,
    val population: String,
    @Json(name = "surface_water") val surfaceWater: String,
    @Json(name = "climate") override val rawClimates: String,
    @Json(name = "terrain") override val rawTerrains: String,
    @Json(name = "residents") override val charactersURLs: List<String>,
    @Json(name = "films") override val filmsURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasCharacters, HasFilms, HasTerrains, HasClimates, ShortDescriptable {
    override val shortDescription: String
        get() = name
}
```

CHARACTER MODEL



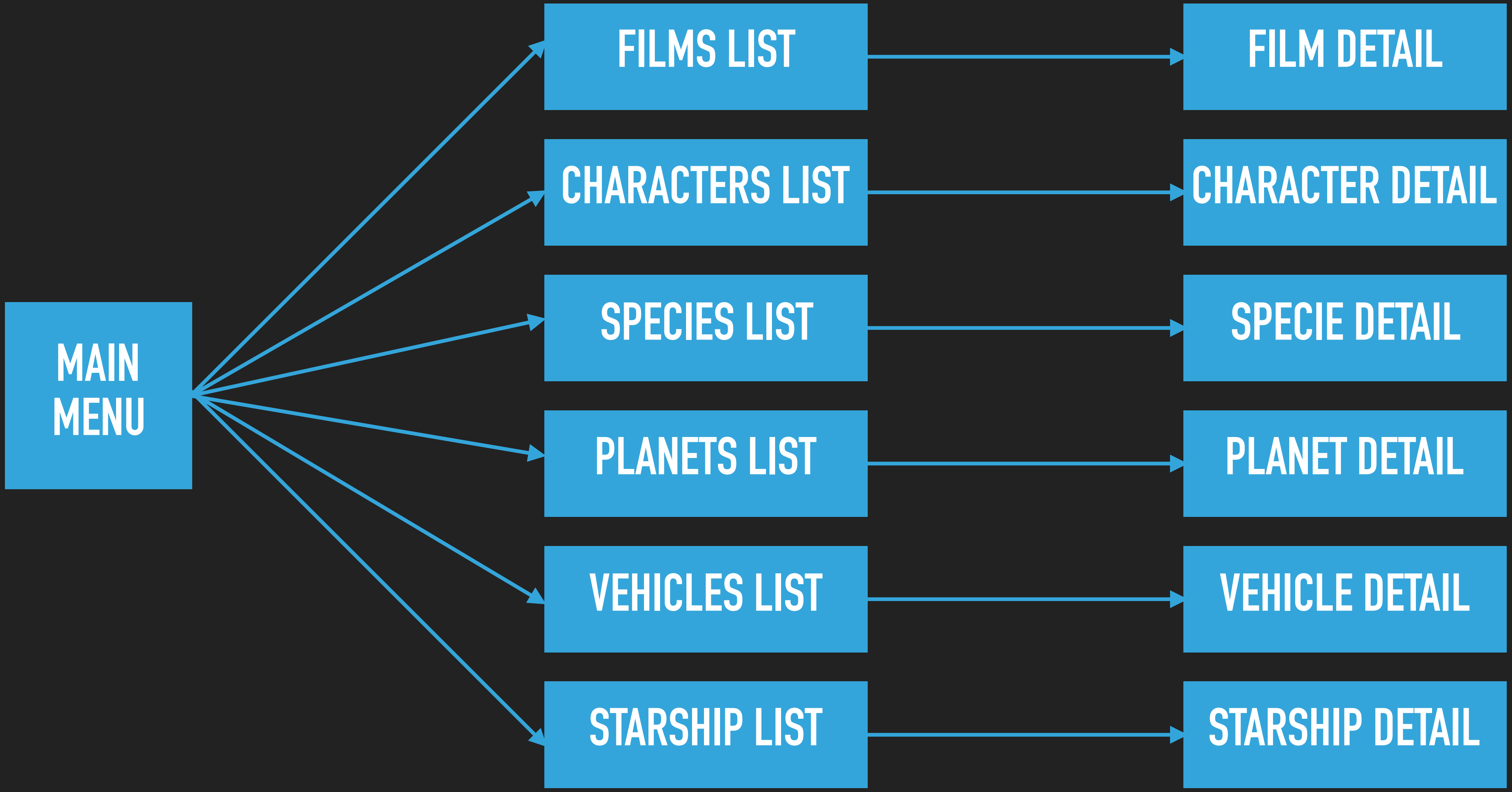
```kotlin
@JsonClass(generateAdapter = true)
@Parcelize
data class Character(
    val name: String,
    val height: String,
    val mass: String,
    @Json(name = "birth_year") val birthYear: String,
    val gender: Gender,
    @Json(name = "hair_color") override val rawHairColors: String,
    @Json(name = "skin_color") override val rawSkinColors: String,
    @Json(name = "eye_color") override val rawEyeColors: String,
    @Json(name = "homeworld") override val homeworldURL: String,
    @Json(name = "films") override val filmsURLs: List<String>,
    @Json(name = "species") override val speciesURLs: List<String>,
    @Json(name = "vehicles") override val vehiclesURLs: List<String>,
    @Json(name = "starships") override val starshipsURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasHomeworld, HasFilms, HasSpecies, HasVehicles, HasStarships,
    HasHairColors, HasSkinColors, HasEyeColors, ShortDescriptable {
    override val shortDescription: String
        get() = name
}
```

SPECIE MODEL



```kotlin
@JsonClass(generateAdapter = true)
@Parcelize
class Specie(
    val name: String,
    val classification: String,
    val designation: String,
    @Json(name = "average_lifespan") val averageLifespan: String,
    @Json(name = "average_height") val averageHeight: String,
    val language: String,
    @Json(name = "skin_colors") override val rawSkinColors: String,
    @Json(name = "hair_colors") override val rawHairColors: String,
    @Json(name = "eye_colors") override val rawEyeColors: String,
    @Json(name = "homeworld") override val homeworldURL: String?,
    @Json(name = "people") override val charactersURLs: List<String>,
    @Json(name = "films") override val filmsURLs: List<String>,
    override val created: OffsetDateTime,
    override val edited: OffsetDateTime,
    override val url: String
) : Parcelable, Identifiable, HasHomeworld, HasCharacters, HasFilms, HasHairColors, HasSkinColors,
    HasEyeColors, ShortDescriptable {
    override val shortDescription: String
        get() = name
}
```

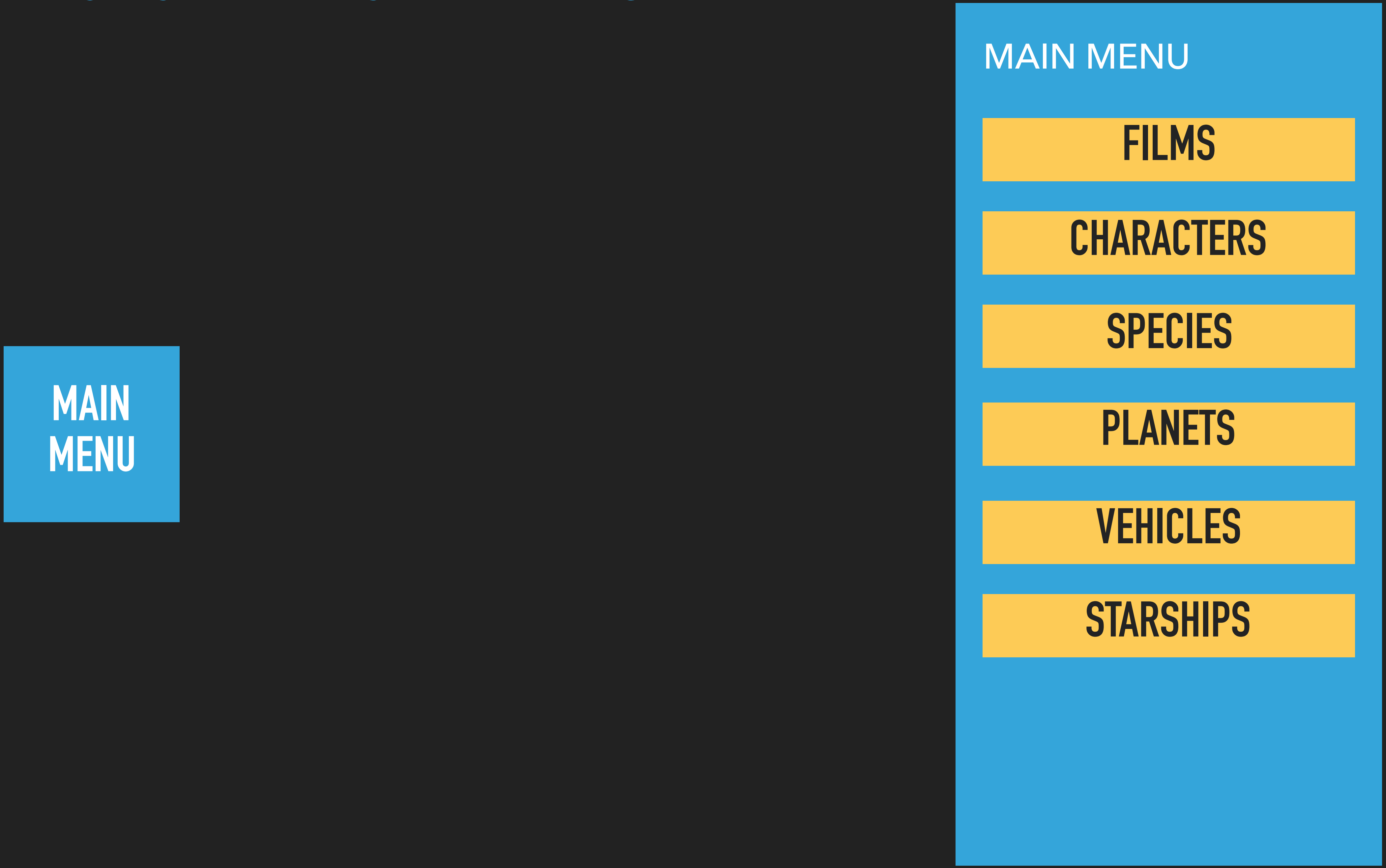
SWAPI MVP APP



13 SCREENS

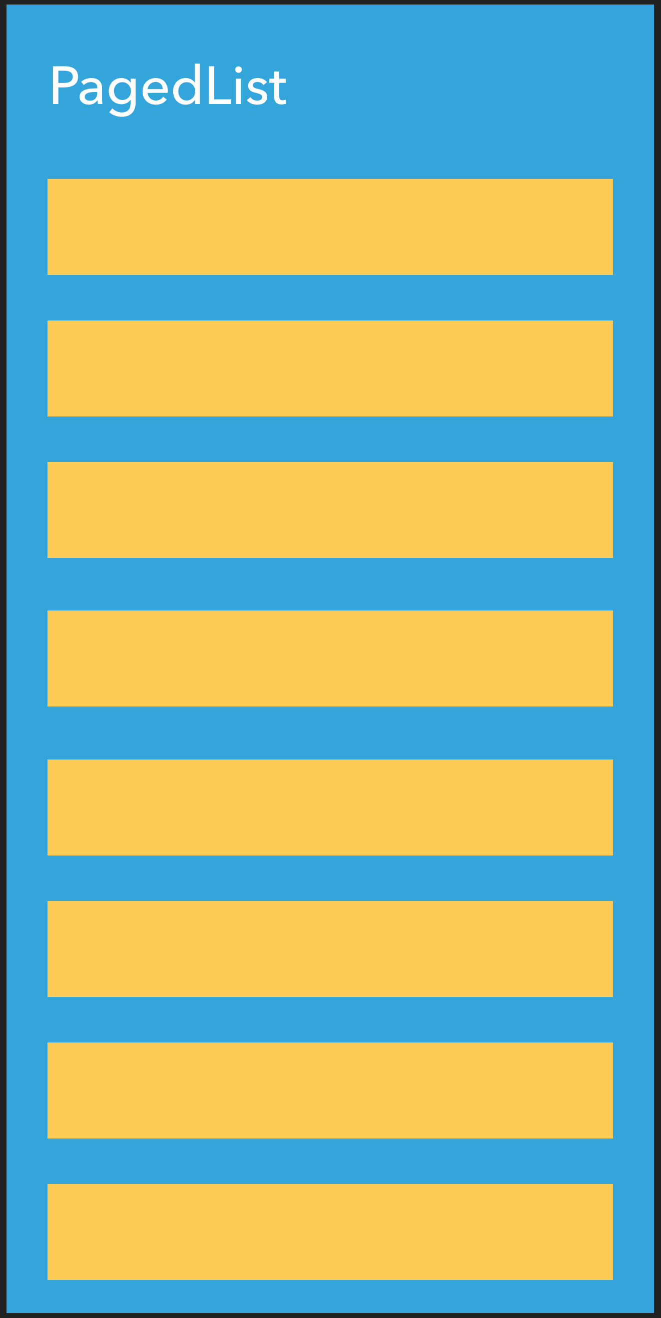
MAIN MENU CAN BE REPRESENTED AS A LIST OF OPTIONS.

EVERY OPTION CAN BE A CELL IN A LIST

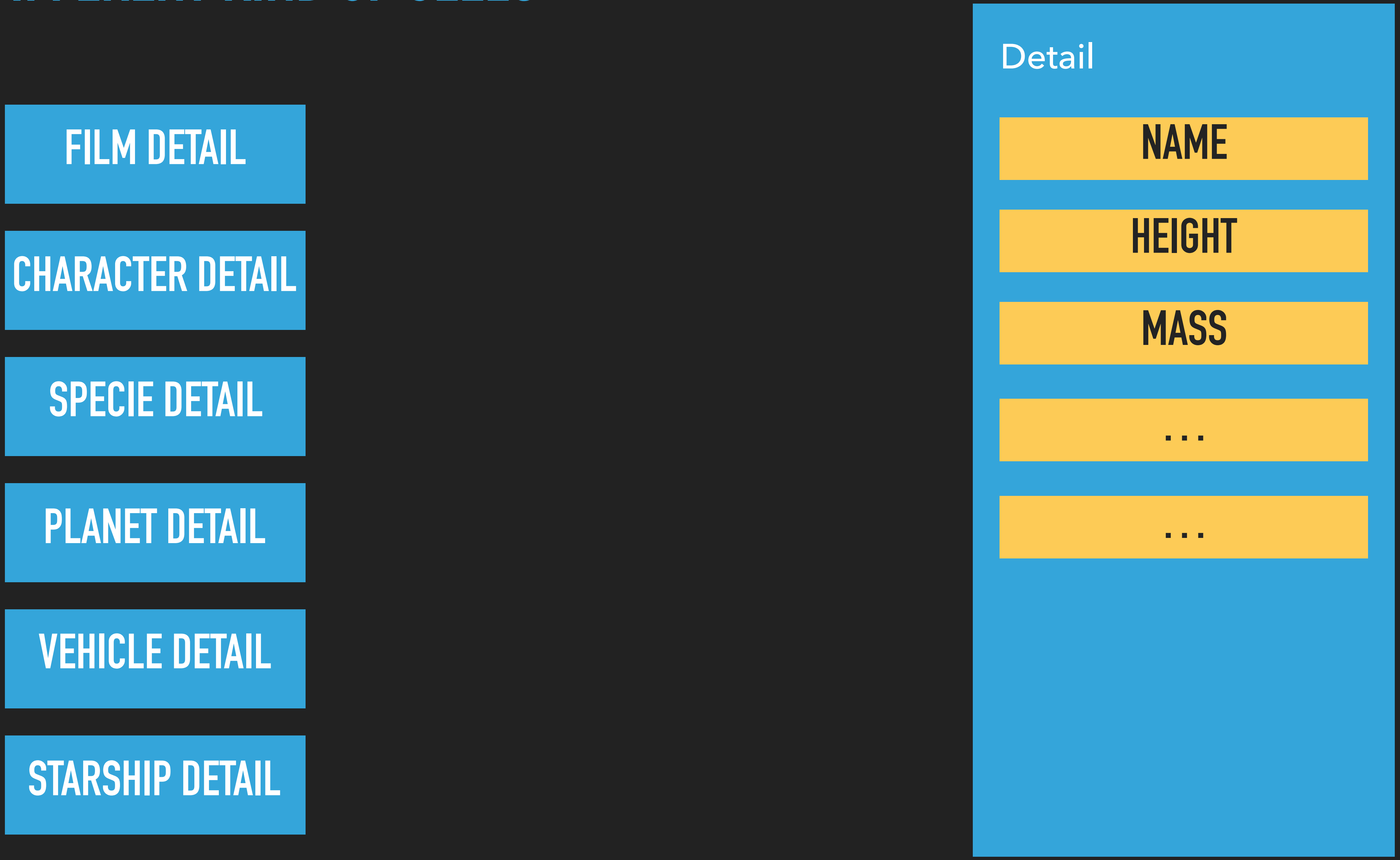


ALL THE LISTS CAN BE REPRESENTED BY A SINGLE FRAGMENT WITH DIFFERENT KIND OF CELLS

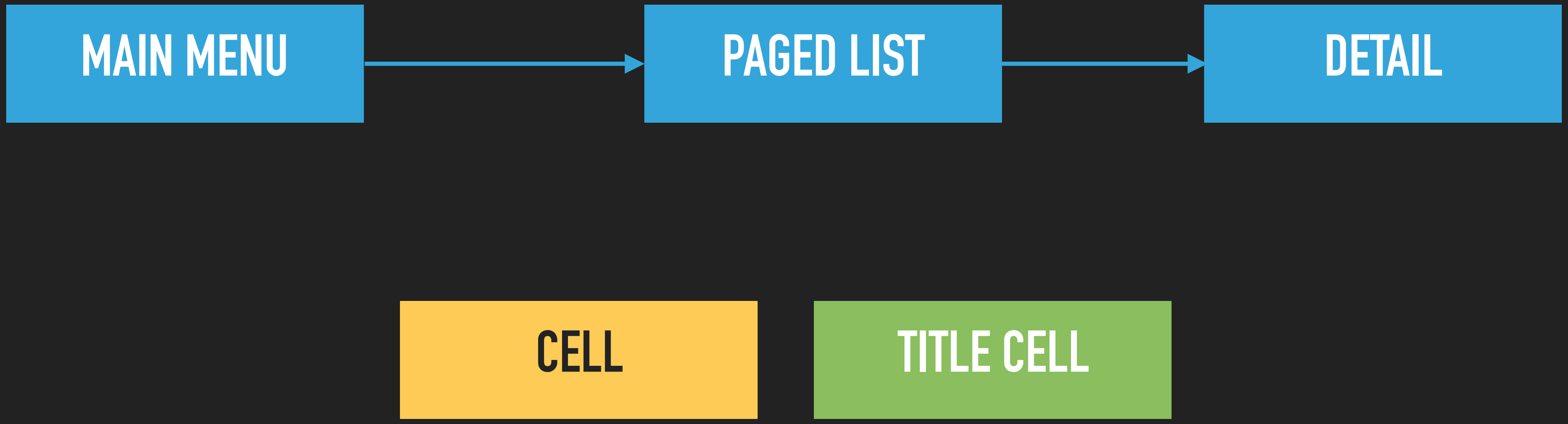
- FILMS LIST
- CHARACTERS LIST
- SPECIES LIST
- PLANETS LIST
- VEHICLES LIST
- STARSHIP LIST



ALL THE DETAILS CAN BE REPRESENTED BY A SINGLE FRAGMENT WITH DIFFERENT KIND OF CELLS



DECLARATIVE/FUNCTIONAL APPROACH



3 SCREENS…
MANY COMPOSABLE & REUSABLE MODELS AND VIEWS

EVERY SCREEN IS A COMPOSITION

LESS CODE, LESS PROBLEMS

🤠

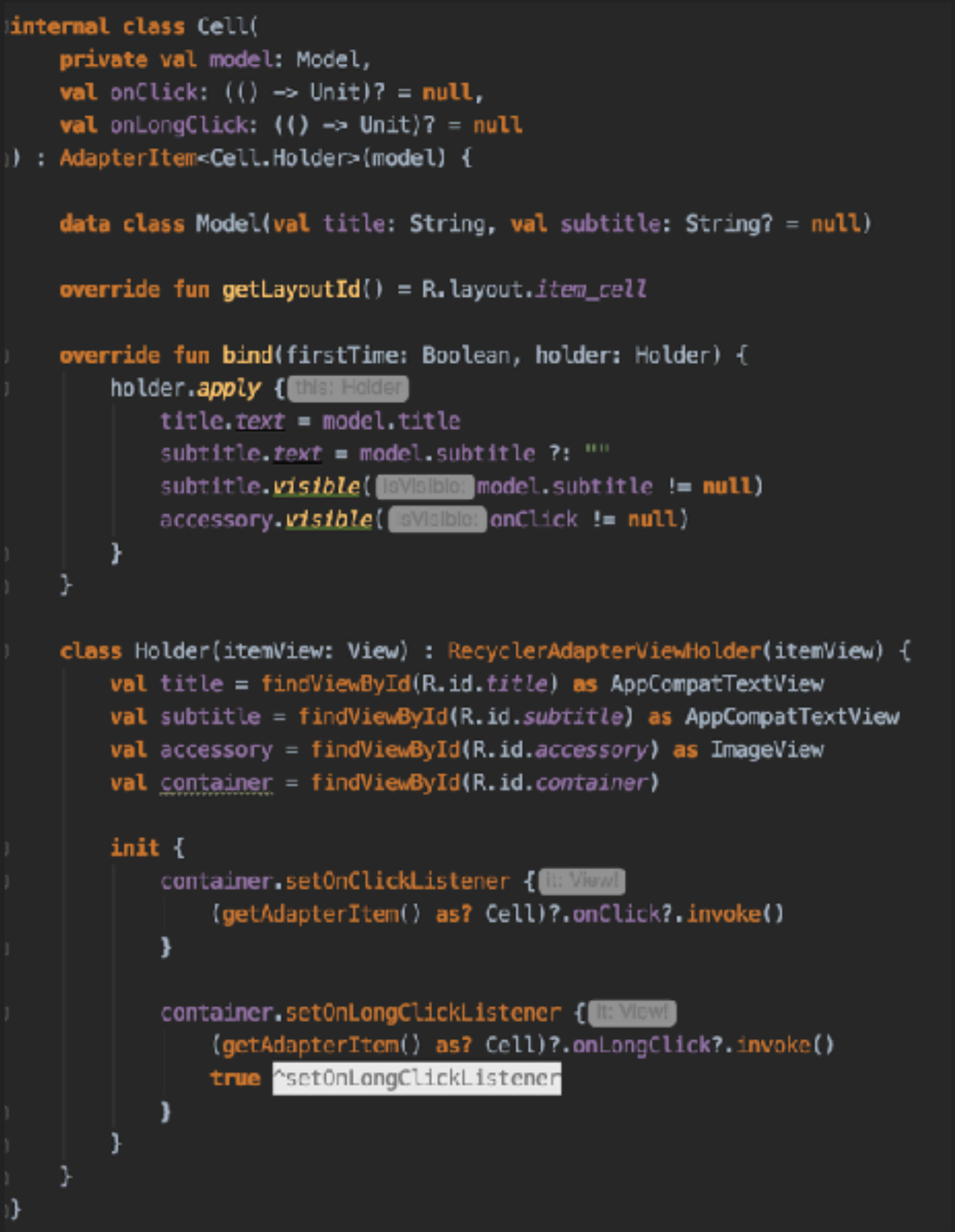
The Lazy Developer

WHAT DO WE NEED?

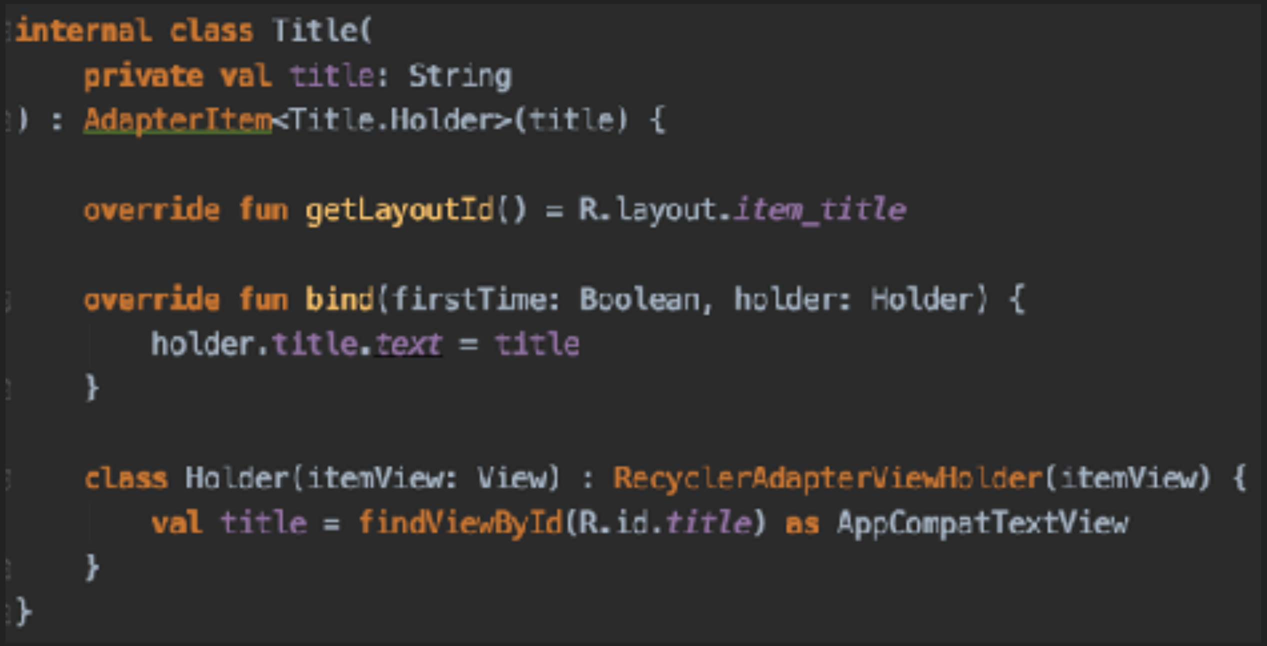
- Recycler View
- Reusable & Composable Cells
- Automatic Diffing

Recycler Adapter

RECYCLER ADAPTER: ADAPTER ITEMS

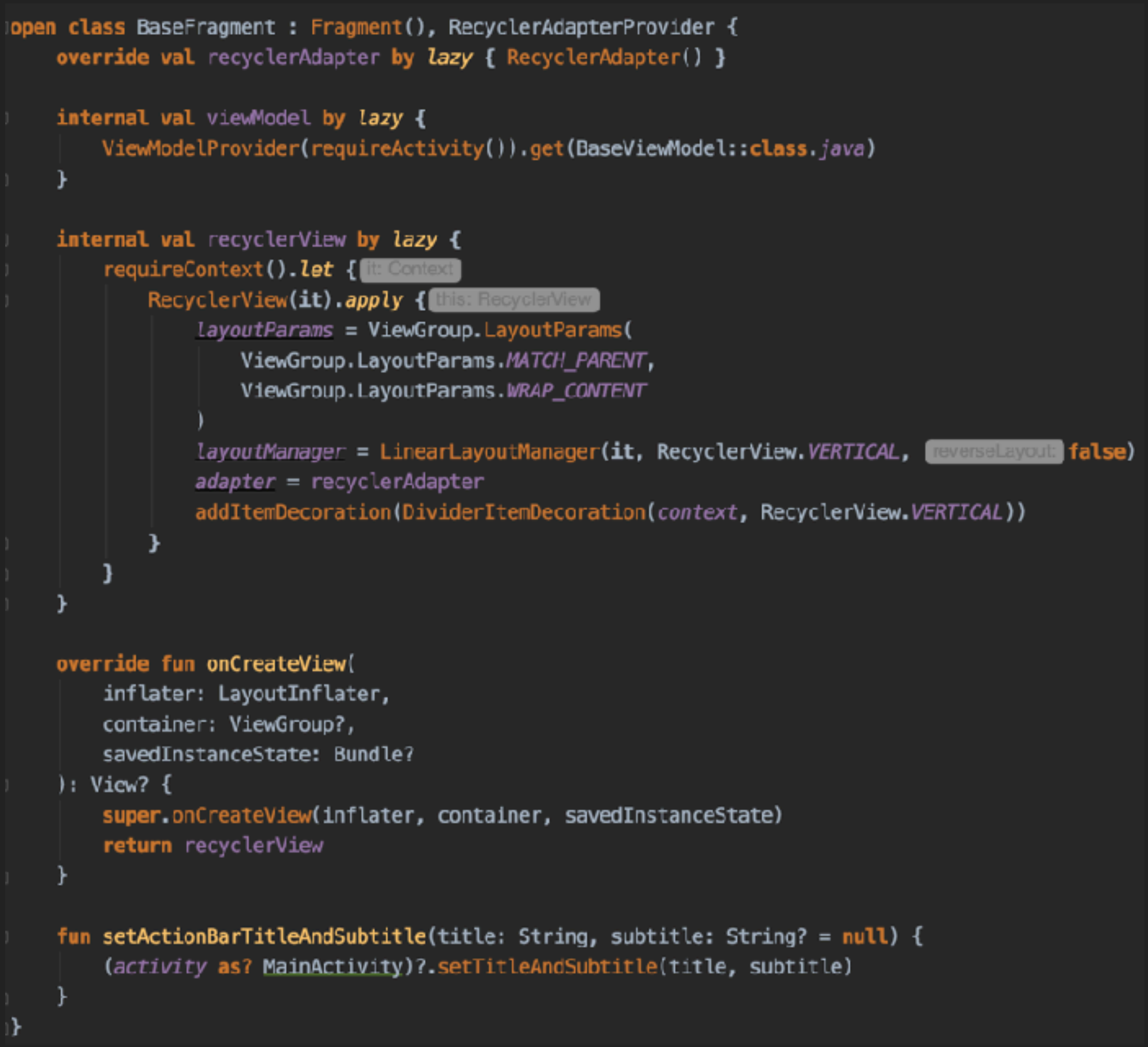


```kotlin
internal class Cell(
    private val model: Model,
    val onClick: (() -> Unit)? = null,
    val onLongClick: (() -> Unit)? = null
) : AdapterItem<Cell.Holder>(model) {

    data class Model(val title: String, val subtitle: String? = null)

    override fun getLayoutId() = R.layout.item_cell

    override fun bind(firstTime: Boolean, holder: Holder) {
        holder.apply { this: Holder
            title.text = model.title
            subtitle.text = model.subtitle ?: ""
            subtitle.visible( isVisible: model.subtitle != null)
            accessory.visible( isVisible: onClick != null)
        }
    }

    class Holder(itemView: View) : RecyclerAdapterViewHolder(itemView) {
        val title = findViewById(R.id.title) as AppCompatTextView
        val subtitle = findViewById(R.id.subtitle) as AppCompatTextView
        val accessory = findViewById(R.id.accessory) as ImageView
        val container = findViewById(R.id.container)

        init {
            container.setOnClickListener { it: View!
                (getAdapterItem() as? Cell)?.onClick?.invoke()
            }

            container.setOnLongClickListener { it: View!
                (getAdapterItem() as? Cell)?.onLongClick?.invoke()
                true ^setOnLongClickListener
            }
        }
    }
}
```



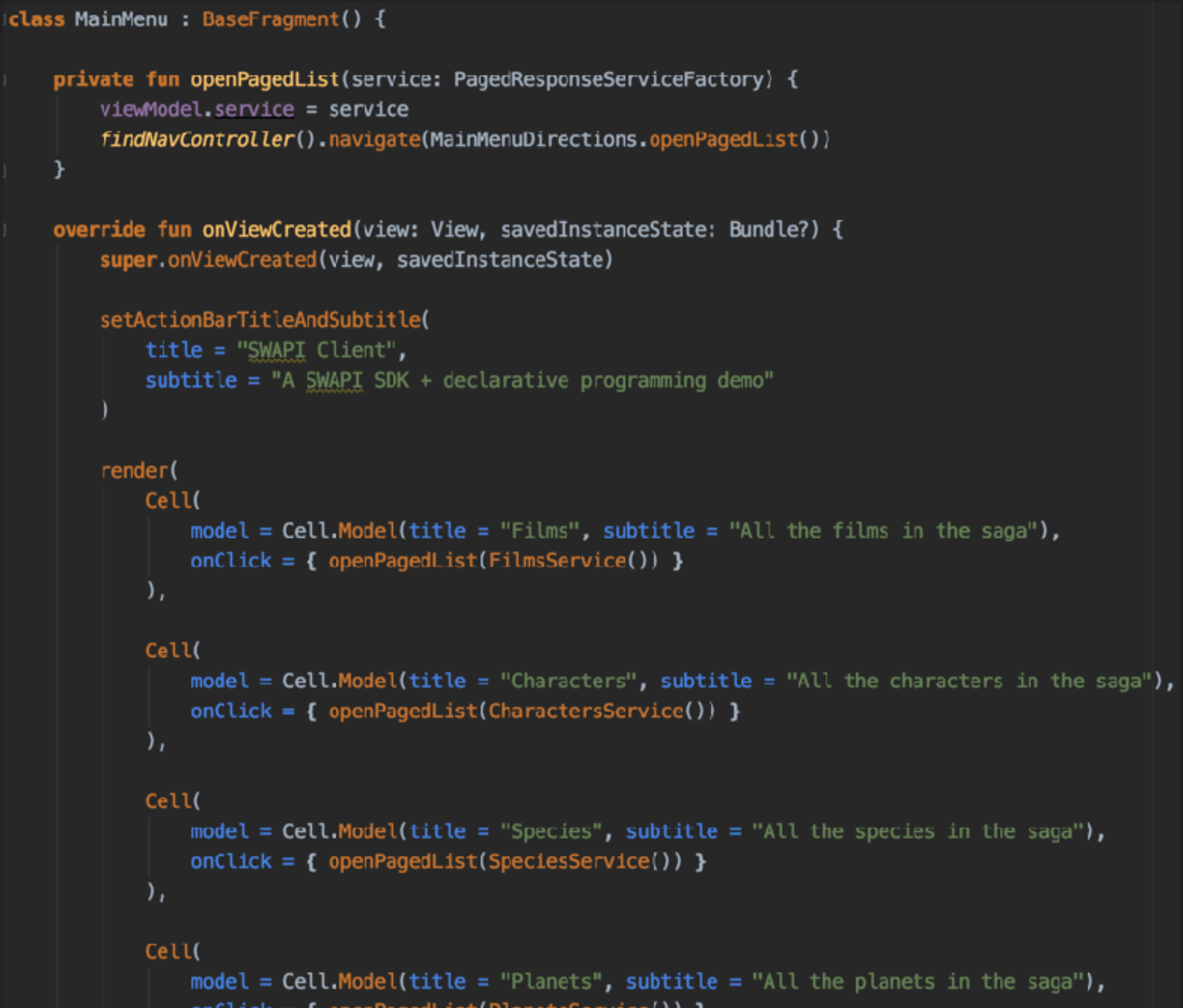
```kotlin
internal class Title(
    private val title: String
) : AdapterItem<Title.Holder>(title) {

    override fun getLayoutId() = R.layout.item_title

    override fun bind(firstTime: Boolean, holder: Holder) {
        holder.title.text = title
    }

    class Holder(itemView: View) : RecyclerAdapterViewHolder(itemView) {
        val title = findViewById(R.id.title) as AppCompatTextView
    }
}
```

BASE FRAGMENT



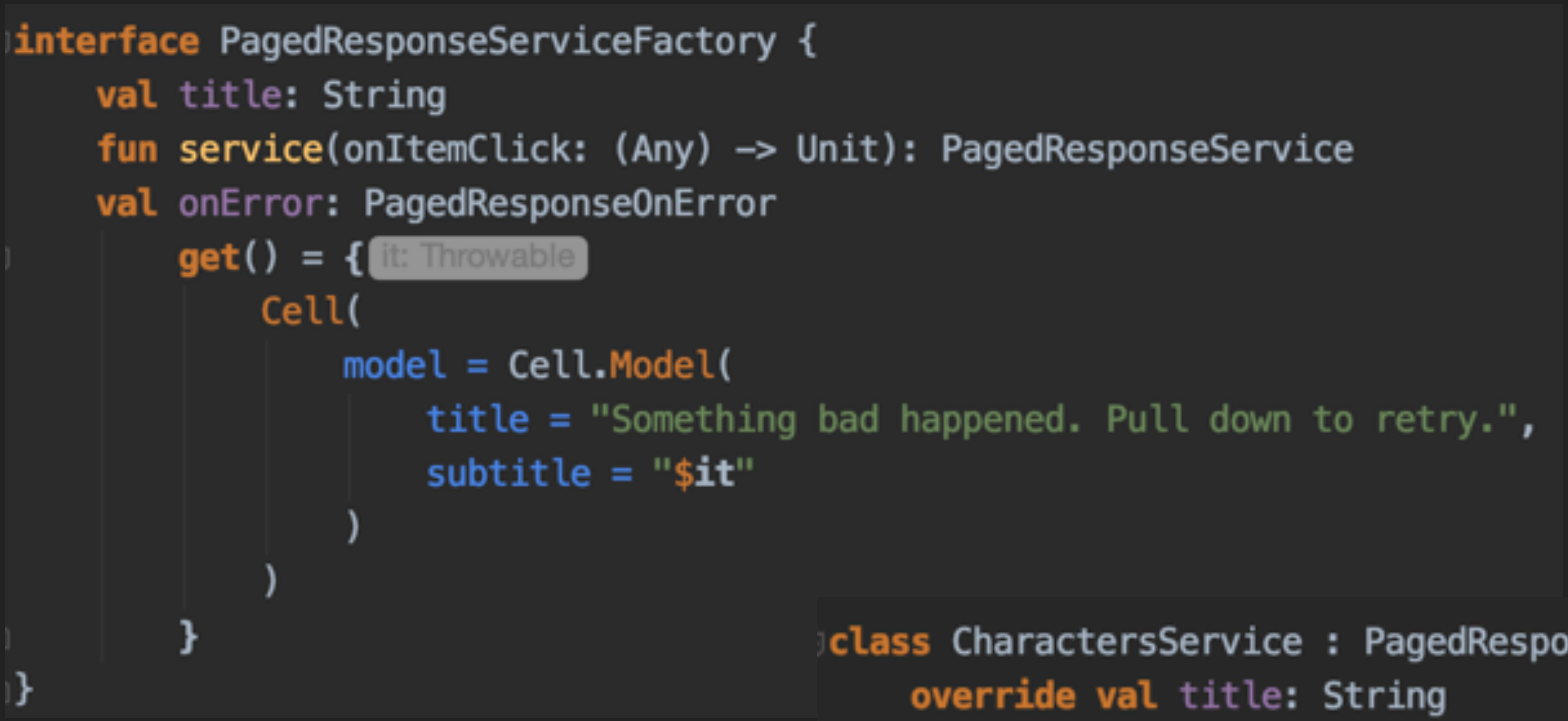
```
open class BaseFragment : Fragment(), RecyclerAdapterProvider {
    override val recyclerAdapter by lazy { RecyclerAdapter() }

    internal val viewModel by lazy {
        ViewModelProvider(requireActivity()).get(BaseViewModel::class.java)
    }

    internal val recyclerView by lazy {
        requireContext().let { it: Context
            RecyclerView(it).apply { this: RecyclerView
                layoutParams = ViewGroup.LayoutParams(
                    ViewGroup.LayoutParams.MATCH_PARENT,
                    ViewGroup.LayoutParams.WRAP_CONTENT
                )
                layoutManager = LinearLayoutManager(it, RecyclerView.VERTICAL,  reverseLayout: false)
                adapter = recyclerAdapter
                addItemDecoration(DividerItemDecoration(context, RecyclerView.VERTICAL))
            }
        }
    }

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        super.onCreateView(inflater, container, savedInstanceState)
        return recyclerView
    }

    fun setActionBarTitleAndSubtitle(title: String, subtitle: String? = null) {
        (activity as? MainActivity)?.setTitleAndSubtitle(title, subtitle)
    }
}
```

THE MAIN MENU

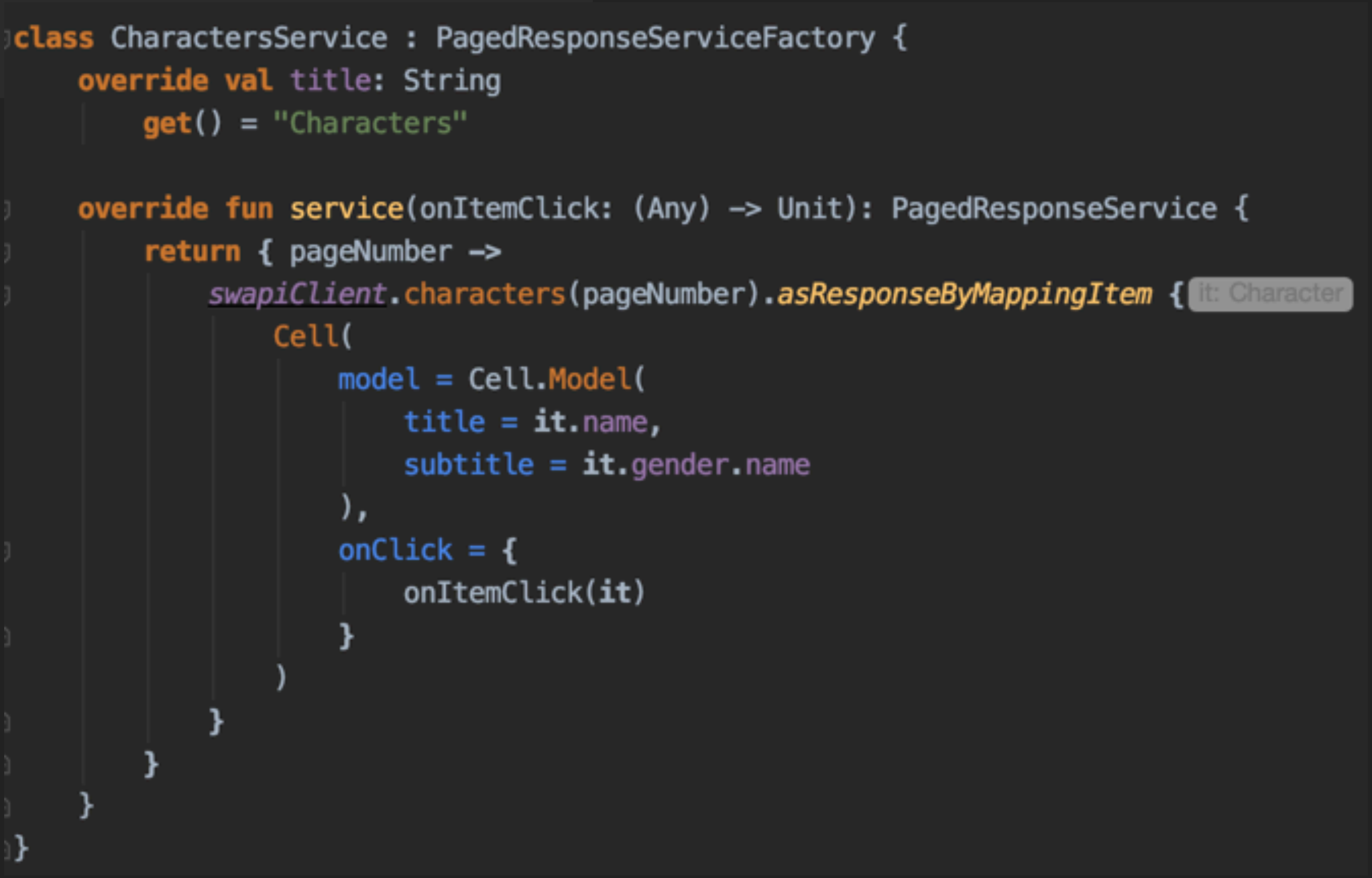


```
class MainMenu : BaseFragment() {

    private fun openPagedList(service: PagedResponseServiceFactory) {
        viewModel.service = service
        findNavController().navigate(MainMenuDirections.openPagedList())
    }

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

        setActionBarTitleAndSubtitle(
            title = "SWAPI Client",
            subtitle = "A SWAPI SDK + declarative programming demo"
        )

        render(
            Cell(
                model = Cell.Model(title = "Films", subtitle = "All the films in the saga"),
                onClick = { openPagedList(FilmsService()) }
            ),

            Cell(
                model = Cell.Model(title = "Characters", subtitle = "All the characters in the saga"),
                onClick = { openPagedList(CharactersService()) }
            ),

            Cell(
                model = Cell.Model(title = "Species", subtitle = "All the species in the saga"),
                onClick = { openPagedList(SpeciesService()) }
            ),

            Cell(
                model = Cell.Model(title = "Planets", subtitle = "All the planets in the saga"),
```

A SERVICE

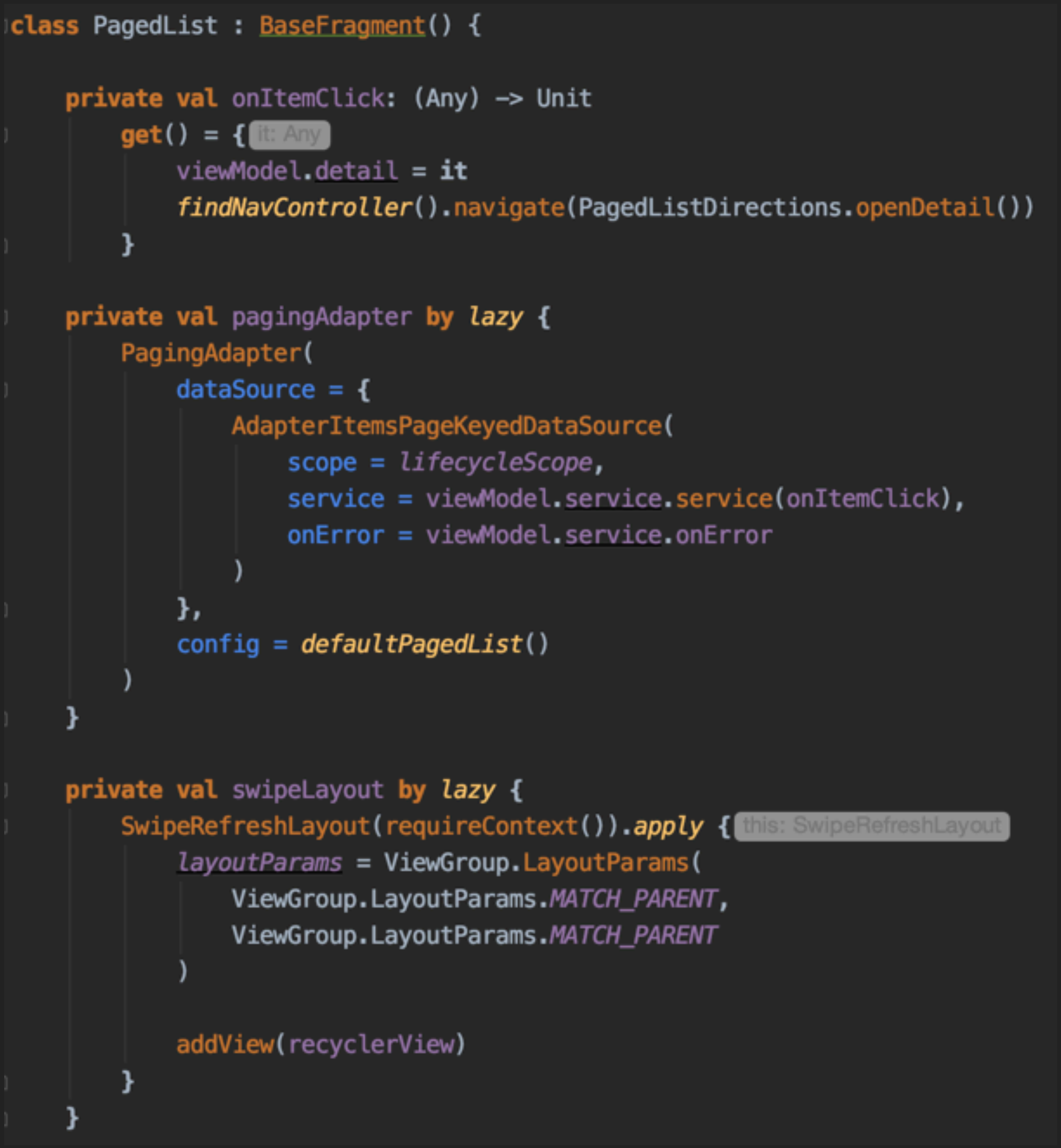


```kotlin
interface PagedResponseServiceFactory {
    val title: String
    fun service(onItemClick: (Any) -> Unit): PagedResponseService
    val onError: PagedResponseOnError
        get() = { it: Throwable
            Cell(
                model = Cell.Model(
                    title = "Something bad happened. Pull down to retry.",
                    subtitle = "$it"
                )
            )
        }
}
```



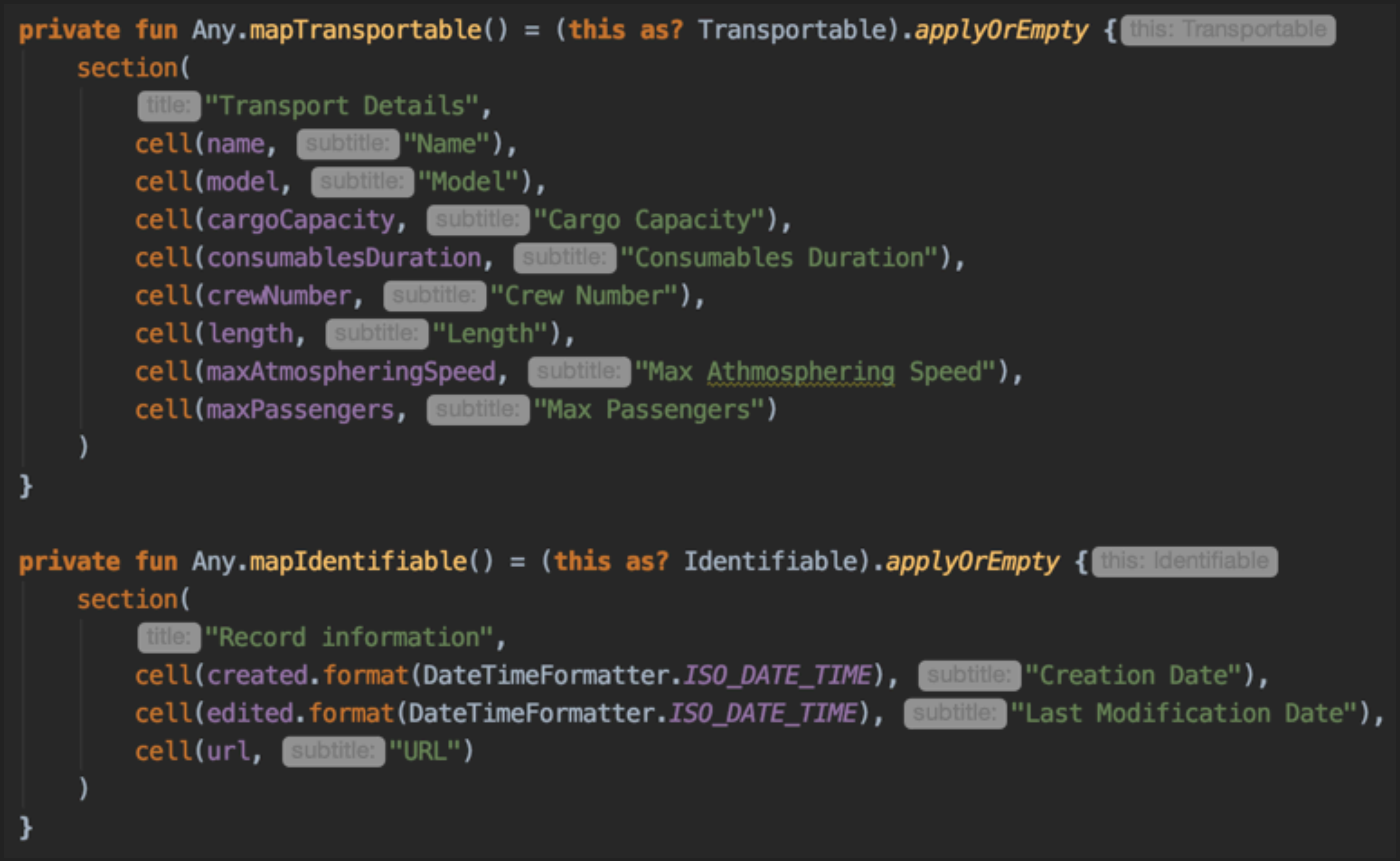
```kotlin
class CharactersService : PagedResponseServiceFactory {
    override val title: String
        get() = "Characters"

    override fun service(onItemClick: (Any) -> Unit): PagedResponseService {
        return { pageNumber ->
            swapiClient.characters(pageNumber).asResponseByMappingItem { it: Character
                Cell(
                    model = Cell.Model(
                        title = it.name,
                        subtitle = it.gender.name
                    ),
                    onClick = {
                        onItemClick(it)
                    }
                )
            }
        }
    }
}
```

PAGED LIST



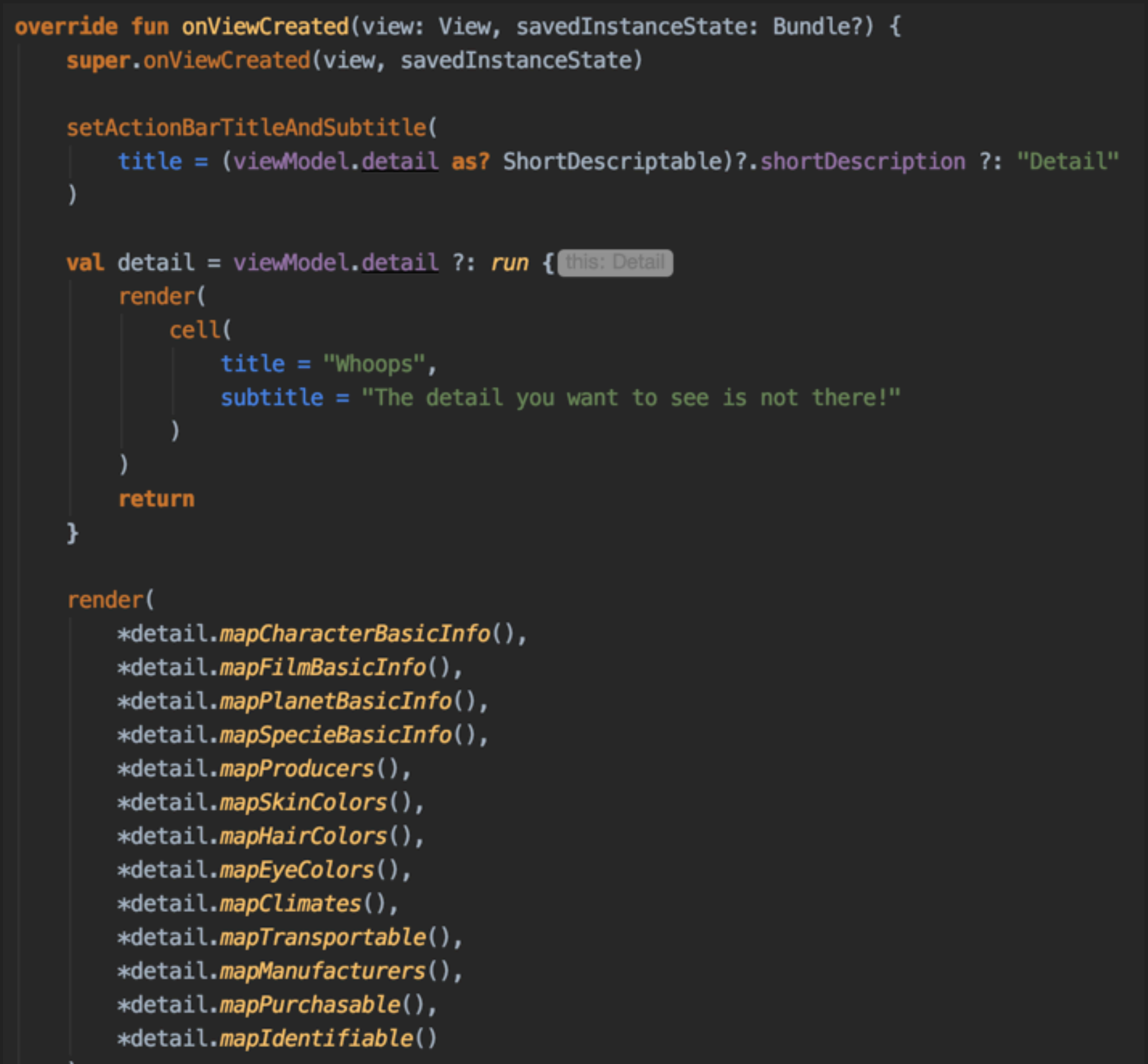
```kotlin
class PagedList : BaseFragment() {

    private val onItemClick: (Any) -> Unit
        get() = {
            viewModel.detail = it
            findNavController().navigate(PagedListDirections.openDetail())
        }

    private val pagingAdapter by lazy {
        PagingAdapter(
            dataSource = {
                AdapterItemsPageKeyedDataSource(
                    scope = lifecycleScope,
                    service = viewModel.service.service(onItemClick),
                    onError = viewModel.service.onError
                )
            },
            config = defaultPagedList()
        )
    }

    private val swipeLayout by lazy {
        SwipeRefreshLayout(requireContext()).apply {
            layoutParams = ViewGroup.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT
            )

            addView(recyclerView)
        }
    }
```

DETAIL



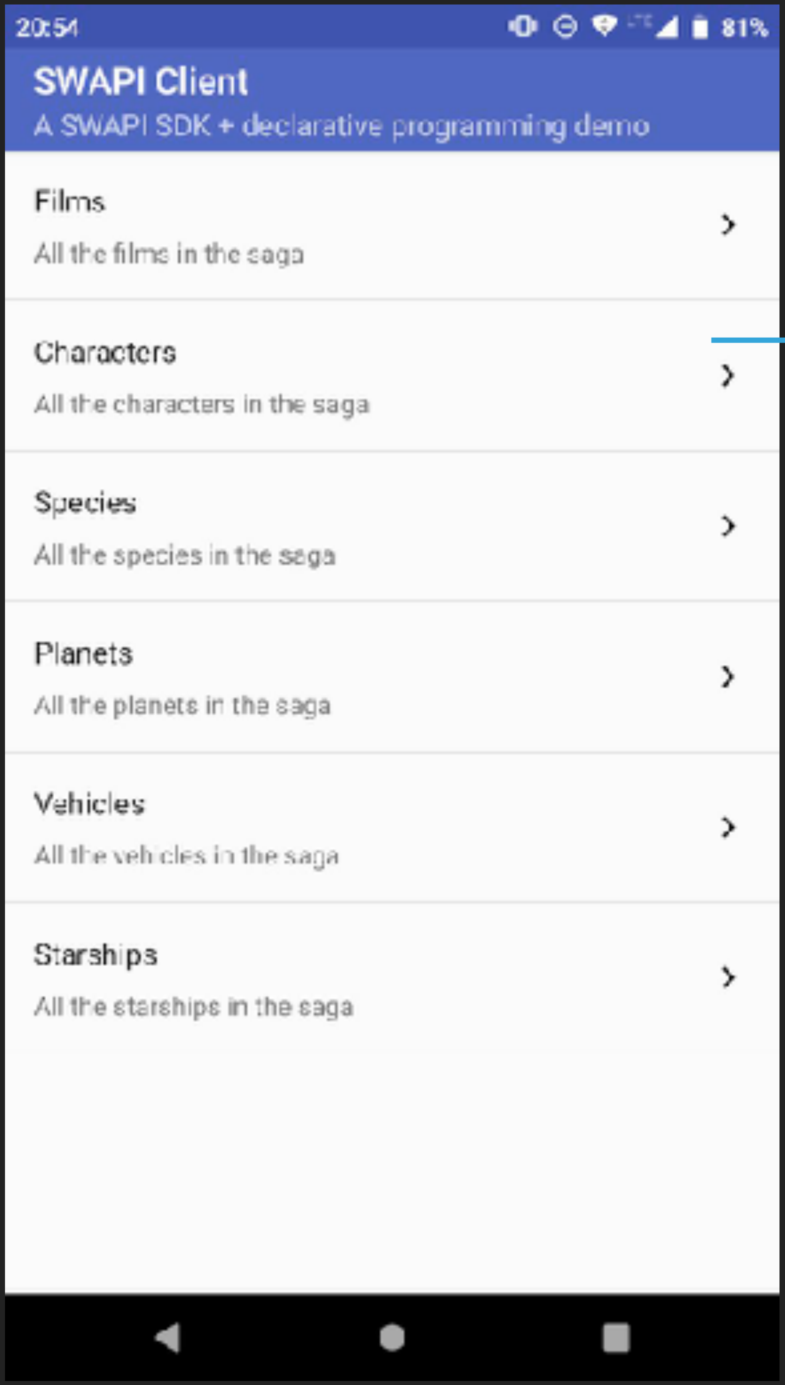
```kotlin
private fun Any.mapTransportable() = (this as? Transportable).applyOrEmpty { this: Transportable
    section(
        title: "Transport Details",
        cell(name, subtitle: "Name"),
        cell(model, subtitle: "Model"),
        cell(cargoCapacity, subtitle: "Cargo Capacity"),
        cell(consumablesDuration, subtitle: "Consumables Duration"),
        cell(crewNumber, subtitle: "Crew Number"),
        cell(length, subtitle: "Length"),
        cell(maxAtmospheringSpeed, subtitle: "Max Athmosphering Speed"),
        cell(maxPassengers, subtitle: "Max Passengers")
    )
}

private fun Any.mapIdentifiable() = (this as? Identifiable).applyOrEmpty { this: Identifiable
    section(
        title: "Record information",
        cell(created.format(DateTimeFormatter.ISO_DATE_TIME), subtitle: "Creation Date"),
        cell(edited.format(DateTimeFormatter.ISO_DATE_TIME), subtitle: "Last Modification Date"),
        cell(url, subtitle: "URL")
    )
}
```

DETAIL



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

    setActionBarTitleAndSubtitle(
        title = (viewModel.detail as? ShortDescriptable)?.shortDescription ?: "Detail"
    )

    val detail = viewModel.detail ?: run { this: Detail
        render(
            cell(
                title = "Whoops",
                subtitle = "The detail you want to see is not there!"
            )
        )
        return
    }

    render(
        *detail.mapCharacterBasicInfo(),
        *detail.mapFilmBasicInfo(),
        *detail.mapPlanetBasicInfo(),
        *detail.mapSpecieBasicInfo(),
        *detail.mapProducers(),
        *detail.mapSkinColors(),
        *detail.mapHairColors(),
        *detail.mapEyeColors(),
        *detail.mapClimates(),
        *detail.mapTransportable(),
        *detail.mapManufacturers(),
        *detail.mapPurchasable(),
        *detail.mapIdentifiable()
    )
```

THE FINAL RESULT



20:54 81%

SWAPI Client

A SWAPI SDK + declarative programming demo

Films
All the films in the saga

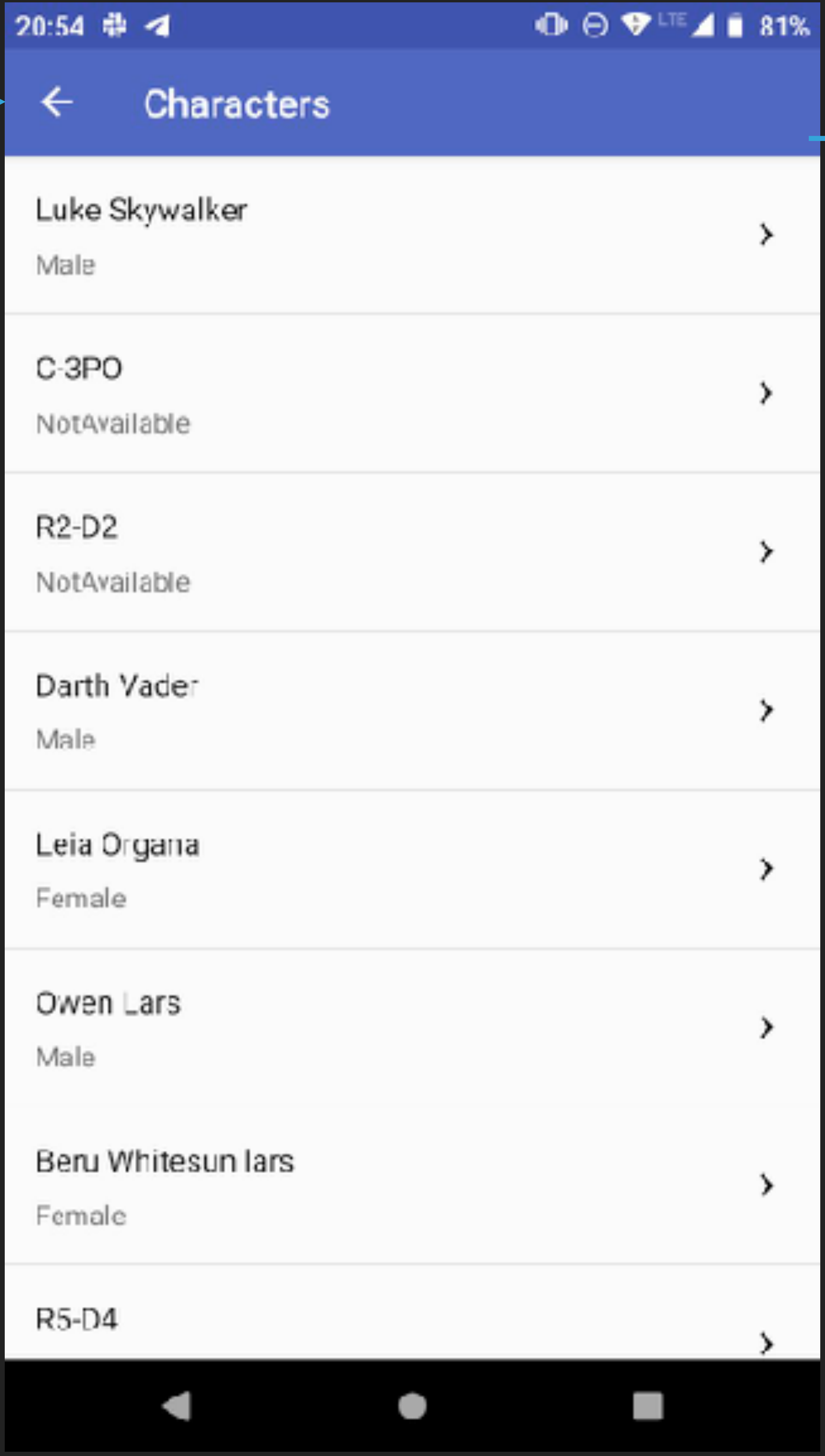
Characters
All the characters in the saga

Species
All the species in the saga

Planets
All the planets in the saga

Vehicles
All the vehicles in the saga

Starships
All the starships in the saga



20:54 81%

Characters

Luke Skywalker
Male

C-3PO
NotAvailable

R2-D2
NotAvailable

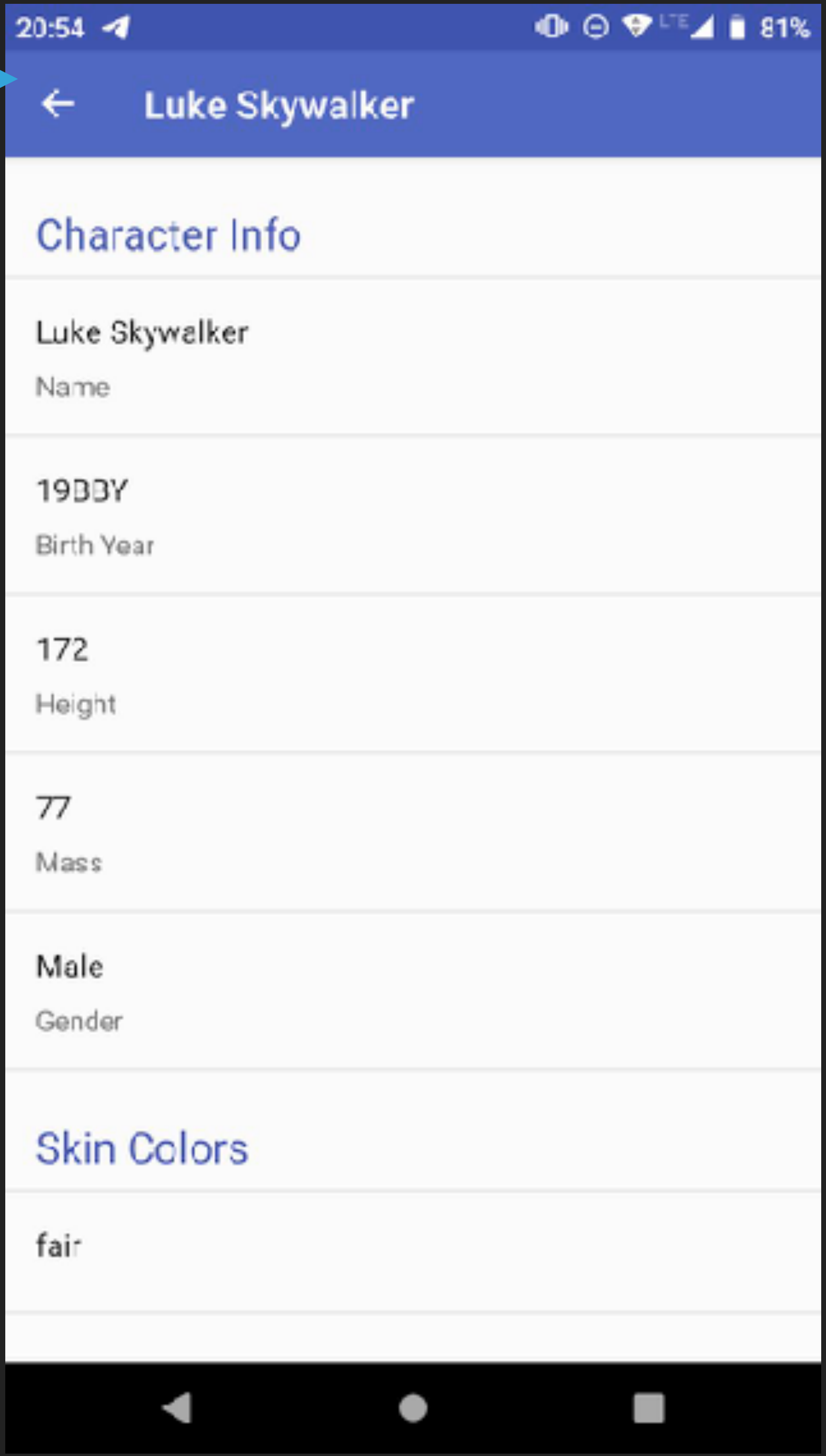
Darth Vader
Male

Leia Organa
Female

Owen Lars
Male

Beru Whitesun lars
Female

R5-D4



20:54 81%

Luke Skywalker

Character Info

Luke Skywalker
Name

19BBY
Birth Year

172
Height

77
Mass

Male
Gender

Skin Colors

fair

THANK YOU!