LİNGODER - DİL ÖĞRENME UYGULAMASI MOBİL PROJE DOKÜMANI

1- PROJE MİMARİSİ (MVVM + Katmanlı Mimari)

View -> ViewModel -> UseCase -> Repository -> NetworkManager

2- PROGRAM AKIŞ ÖRNEK

SignUpView -> Kullanıcının girdiği veriler ile viewmodel tetiklenir.

SignUpViewModel -> SignUpView'dan gelen verileri işler, UseCase çağırır ve SignUpView'ı günceller.

SignUpUseCase -> Repository katmanına giriş isteği atar.

AuthRepository -> NetworkManager ile iletişime geçer.

NetworkManager -> API isteklerini yönetir.

3- PROGRAM AKIŞ ÖRNEK KOD

Dosya Yapısı

```
view/
 ├── SignUpView.kt        → Kullanıcıdan ad, email, şifre alır (Jetpack Compose)
 └── LoginView.kt         → Kullanıcıdan email, şifre alır ve token alır (Jetpack Compose)

viewmodel/
 ├── SignUpViewModel.kt     → SignUpView ile UseCase arasında köprü
 └── LoginViewModel.kt      → LoginView ile UseCase arasında köprü + token'ı bellekte tutar

domain/
 ├── usecase/
 |    ├── SignUpUseCase.kt    → Kayıt olma mantığı
 |    └── LoginUseCase.kt     → Giriş yapma mantığı
 └── repository/
      └── AuthRepository.kt   → API'ye erişimi sağlayan soyutlama katmanı
```

```
data/
 └──── network/
    └──── NetworkManager.kt    → Tüm GET, POST, PUT, DELETE işlemleri burada yönetilir (JWT destekli)

model/
 ├──── SignUpRequest.kt        → Ad, email, şifre içeren kayıt modeli
 ├──── LoginRequest.kt        → Email, şifre içeren giriş modeli
 └──── LoginResponse.kt        → Backend'den dönen JWT token

MainActivity.kt             → Uygulamanın başlangıç noktası, ekranlar buradan tetiklenir
```

```
data class SignUpRequest(
    val name: String,
    val email: String,
    val password: String
)
```

-> MODEL

```
data class LoginRequest(
    val email: String,
    val password: String
)
data class LoginResponse(
    val token: String
)

class NetworkManager(private val jwtToken: String? = null) {

    inline fun <reified T> get(endpoint: String): T? {
        val url = URL(endpoint)
        val connection = url.openConnection() as HttpURLConnection
        connection.requestMethod = "GET"
        setHeaders(connection)

        return if (connection.responseCode in 200..299) {
            Gson().fromJson(connection.inputStream.reader(), T::class.java)
        } else {
            null

}
}
    inline fun <reified T> postForResult(endpoint: String, body: Any): T? {
        val url = URL(endpoint)
        val connection = url.openConnection() as HttpURLConnection
        connection.requestMethod = "POST"
        connection.doOutput = true
        setHeaders(connection)

        val json = Gson().toJson(body)
        connection.outputStream.write(json.toByteArray())

        return if (connection.responseCode in 200..299) {
            Gson().fromJson(connection.inputStream.reader(), T::class.java)
        } else null
    }
```

-> NetworkManager

```
    fun post(endpoint: String, body: Any): Boolean {
        val url = URL(endpoint)
        val connection = url.openConnection() as HttpURLConnection
        connection.requestMethod = "POST"
        connection.doOutput = true
        setHeaders(connection)

        val json = Gson().toJson(body)
        connection.outputStream.write(json.toByteArray())

        return connection.responseCode in 200..299
    }

    fun put(endpoint: String, body: Any): Boolean {
        val url = URL(endpoint)
        val connection = url.openConnection() as HttpURLConnection
        connection.requestMethod = "PUT"
        connection.doOutput = true
        setHeaders(connection)

        val json = Gson().toJson(body)
        connection.outputStream.write(json.toByteArray())

        return connection.responseCode in 200..299
    }

    fun delete(endpoint: String): Boolean {
        val url = URL(endpoint)
        val connection = url.openConnection() as HttpURLConnection
        connection.requestMethod = "DELETE"
        setHeaders(connection)

        return connection.responseCode in 200..299
    }

    private fun setHeaders(connection: HttpURLConnection) {
        connection.setRequestProperty("Content-Type", "application/json")
        jwtToken?.let {
            connection.setRequestProperty("Authorization", "Bearer $it")
        }
    }
}

class AuthRepository(private val networkManager: NetworkManager) {

    fun signUp(request: SignUpRequest): Boolean {
        return networkManager.post("http://localhost:8080/auth/signup", request)
    }
```

-> AuthRepository

```
   fun login(email: String, password: String): String? {
        val request = LoginRequest(email, password)
        val response: LoginResponse? = networkManager.postForResult("http://localhost:8080/auth/login", request)
        return response?.token
    }

    // fun getProfile(): User? = networkManager.get("http://localhost:8080/user/profile")
}


class SignUpUseCase(private val repository: AuthRepository) {
    fun execute(request: SignUpRequest): Boolean {
        return repository.signUp(request)
    }
```

-> UseCase

```
}
class LoginUseCase(private val repository: AuthRepository) {
    fun execute(email: String, password: String): String? {
        return repository.login(email, password)
    }
```

```
}




class SignUpViewModel(private val useCase: SignUpUseCase) {

    fun signUp(name: String, email: String, password: String): Boolean {

        val request = SignUpRequest(name, email, password)

        return useCase.execute(request)

    }

}

class LoginViewModel(

    private val loginUseCase: LoginUseCase,

    private val networkManager: NetworkManager

) {

    var token: String? = null

        private set


    fun login(email: String, password: String): Boolean {

        val resultToken = loginUseCase.execute(email, password)

        return if (resultToken != null) {

            token = resultToken

            networkManager.updateToken(resultToken)

            true

        } else {

            false

        }

    }

}
```

->ViewModel

```
@Composable
fun SignUpScreen(viewModel: SignUpViewModel) {
    var name by remember { mutableStateOf("") }
    var email by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var message by remember { mutableStateOf<String?>(null) }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(24.dp),
        verticalArrangement = Arrangement.Center
    ) {
        Text(text = "Sign Up", style = MaterialTheme.typography.headlineSmall)

        Spacer(modifier = Modifier.height(16.dp))
```

->View

```
        OutlinedTextField(
            value = name,
            onValueChange = { name = it },
            label = { Text("Ad") },
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(modifier = Modifier.height(8.dp))

        OutlinedTextField(
            value = email,
            onValueChange = { email = it },
            label = { Text("Email") },
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(modifier = Modifier.height(8.dp))

        OutlinedTextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Şifre") },
            visualTransformation = PasswordVisualTransformation(),
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(modifier = Modifier.height(16.dp))

        Button(
            onClick = {
                val result = viewModel.signUp(name, email, password)
                message = if (result) "      Kayıt başarılı" else "      Kayıt başarısız"
            },
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Kayıt Ol")
        }

        message?.let {
            Spacer(modifier = Modifier.height(12.dp))
            Text(text = it)
        }
    }
}
```

@Composable

```
fun LoginScreen(viewModel: LoginViewModel) {
    var email by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var loginMessage by remember { mutableStateOf<String?>(null) }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(24.dp),
        verticalArrangement = Arrangement.Center
    ) {
        Text("Login", style = MaterialTheme.typography.headlineSmall)
        Spacer(Modifier.height(16.dp))

        OutlinedTextField(
            value = email,
            onValueChange = { email = it },
            label = { Text("Email") },
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(Modifier.height(8.dp))

        OutlinedTextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Password") },
            visualTransformation = PasswordVisualTransformation(),
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(Modifier.height(16.dp))
        Button(
            onClick = {
                val success = viewModel.login(email, password)
                loginMessage = if (success) "      Giriş başarılı! Token alındı." else "      Giriş başarısız."
            },
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Giriş Yap")
        }

        loginMessage?.let {
            Spacer(Modifier.height(12.dp))
            Text(it)
        }
    }
}
```

-> MainActivity

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        val networkManager = NetworkManager()
        val repository = AuthRepository(networkManager)
        val useCase = SignUpUseCase(repository)
        val viewModel = SignUpViewModel(useCase)

        setContent {
            SignUpScreen(viewModel)
        }
```

```
  }
}
```

SignUpScreen (View)

  ↓

SignUpViewModel

  ↓

SignUpUseCase

  ↓

AuthRepository

  ↓

NetworkManager → API'ye POST /auth/signup

1. Genel Bilgiler

- **İletişim Protokolü: HTTP/HTTPS**
- **Veri Formatı: JSON (UTF-8)**
- **Kimlik Doğrulama:**
  - `POST /auth/login` **endpoint'i başarılı şekilde çağrıldığında JWT token döner.**
  - **JWT Token,** `"Authorization: Bearer <token>"` **şeklinde header'da iletilmelidir.**
  - **Token gerektirmeyen endpoint'ler:**
    - `POST /auth/signup`
    - `POST /auth/login`
    - `POST /auth/forgot`
    - `POST /auth/reset`
  - **Token gerektiren örnek endpoint'ler:**
    - `GET /word`
    - `POST /word/add`

2. Endpoint’ler

2.1. Kullanıcı Kaydı

Endpoint: `POST /auth/signup`

- **Açıklama: Yeni kullanıcı kaydı oluşturur.**

İstek Başlığı (Headers):

```
Content-Type: application/json
```

İstek Gövdesi (Body):

```
{
  "username": "YusufDemir",
  "email": "yusuf@example.com",
  "password": "sifre123",
  “role”: “USER”
}
```

Başarılı Yanıt:

```
Status: 200 OK
```

Hatalı Yanıt (Örnek):

```
{
  "error": "User already exists"
}
```

2.2. Giriş Yap

Endpoint: `POST /auth/signin`

- **Açıklama: Kullanıcı adı ve şifre ile giriş yapar ve JWT token alır.**

İstek Başlığı (Headers):

```
Content-Type: application/json
```

İstek Gövdesi (Body):

```
{
  "email": "yusuf@example.com",
  "password": "sifre123"
}
```

Başarılı Yanıt:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR..."
}
```

2.3. Şifre Sıfırlama

Endpoint: `POST /auth/forgot`

- **Açıklama: Kullanıcı e-posta ile doğrulama kodu alır.**

İstek Başlığı (Headers):

```
Content-Type: application/json
```

İstek Gövdesi (Body):

```
{
  "email": "yusuf@example.com"
}
```

Başarılı Yanıt:

```
Status: 200 OK
```

2.3.1 Şifre Sıfırlama

Endpoint: `POST /auth/reset`

- **Açıklama: Yeni kullanıcı kaydı oluşturur.**

İstek Başlığı (Headers):

```
Content-Type: application/json
```

İstek Gövdesi (Body):

```
{
  "code": “965217”,
  "newPassword": "sifre123456"
}
```

Başarılı Yanıt:

```
Status: 200 OK
```

Hatalı Yanıt (Örnek):

```
{
  "error": "Code not found"
}
```

2.4 Kelime Listeleme

Endpoint: `GET /word`

- **Açıklama: Kullanıcı JWT Token ile kelimelerini görür.**

İstek Başlığı (Headers):

```
Authorization: Bearer <token>
```

Başarılı Yanıt:

```
{
  "id": "bbfdaed5-8dfe-4702-bfdd-294a425efc64",
  “topic”: “Fruits”,
  “engWordName”: “Apple”,
  “turkWordName”: “Elma”,
  “sentences”: “This apple is so sweet.”,
  “accuracy”: 0.0,
  “image”: “https://example.com/apple.jpg”
},
{
  "id": "bbfdaed5-8dfe-4702-bfdd-294a425efc64",
  “topic”: “ANIMALS”,
  “engWordName”: “Lion”,
  “turkWordName”: “Aslan”,
  “sentences”: “This lion is so angry.”,
  “accuracy”: 0.0,
  “image”: “https://example.com/lion.jpg”
}
```

2.4.1 Kelime Ekleme

Endpoint: `POST /word/add`

- **Açıklama: Kullanıcı JWT Token ile kelime ekler.**

İstek Başlığı (Headers):

```
Authorization: Bearer <token>
```

İstek Gövdesi (Body):

```
{
  "username": "YusufDemir",
  “engWordName”: “Apple”,
  “turkWordName”: “Elma”,
  “sentences”: “This apple is so sweet.”,
  “picture”: “https://example.com/apple.jpg”
  “topic”: “Fruits”
}
```

Başarılı Yanıt:

```
{
  "id": "bbfdaed5-8dfe-4702-bfdd-294a425efc64",
}
```

2.4.2 Kelime Silme

Endpoint: `DELETE /word/delete/{id}`

- **Açıklama: Kullanıcı JWT Token ile kelimeyi siler.**

İstek Başlığı (Headers):

```
Authorization: Bearer <token>
```

Başarılı Yanıt:

```
Status: 200 OK
```