Go Fejlesztő - Állásinterjú Gyakorlati Feladat

Feladat Áttekintés

Időkeret: 2 óra
Technológiák: Go, REST API, ORM (GORM), Docker, PostgreSQL
Cél: Egy egyszerű könyvtár kezelő rendszer backend API-jának elkészítése

---

Projekt Követelmények

1. Alapbeállítások (15 perc)

- Go projekt inicializálása go mod-dal
- Szükséges dependencies telepítése
- Projekt struktúra létrehozása

2. Adatbázis Modell (20 perc)

Implementáld a következő entitásokat GORM használatával:

Book Model

```
type Book struct {
    ID          uint      `json:"id" gorm:"primaryKey"`
    Title       string    `json:"title" gorm:"not null"`
    Author      string    `json:"author" gorm:"not null"`
    ISBN        string    `json:"isbn" gorm:"unique;not null"`
    Year        int       `json:"year"`
    Available   bool      `json:"available" gorm:"default:true"`
    CreatedAt   time.Time `json:"created_at"`
    UpdatedAt   time.Time `json:"updated_at"`
}
```

Borrowing Model

```
type Borrowing struct {
    ID          uint      `json:"id" gorm:"primaryKey"`
    BookID      uint      `json:"book_id"`
    Book        Book      `json:"book" gorm:"foreignKey:BookID"`
    BorrowerName string   `json:"borrower_name" gorm:"not null"`
    BorrowDate  time.Time `json:"borrow_date"`
    ReturnDate  *time.Time `json:"return_date"`
    CreatedAt   time.Time `json:"created_at"`
    UpdatedAt   time.Time `json:"updated_at"`
}
```

3. REST API Endpoints (45 perc)

Implementáld a következő HTTP endpoint-okat:

Books Management

- `GET /api/books` - Összes könyv listázása
- `GET /api/books/:id` - Egy könyv részletei
- `POST /api/books` - Új könyv létrehozása
- `PUT /api/books/:id` - Könyv adatainak frissítése
- `DELETE /api/books/:id` - Könyv törlése

Borrowing Management

- `POST /api/books/:id/borrow` - Könyv kölcsönzése
- `POST /api/books/:id/return` - Könyv visszahozatala
- `GET /api/borrowings` - Aktív kölcsönzések listája

Search & Filter

- `GET /api/books?search=title` - Könyv keresés cím alapján
- `GET /api/books?available=true` - Csak elérhető könyvek

4. Request/Response Struktúrák (15 perc)

Create Book Request

```
type CreateBookRequest struct {
    Title  string `json:"title" binding:"required"`
    Author string `json:"author" binding:"required"`
    ISBN   string `json:"isbn" binding:"required"`
    Year   int    `json:"year" binding:"required,min=1900,max=2024"`
}
```

Borrow Request

```
type BorrowRequest struct {
    BorrowerName string `json:"borrower_name" binding:"required"`
}
```

Error Response

```
type ErrorResponse struct {
    Error   string `json:"error"`
    Message string `json:"message,omitempty"`
}
```

5. Docker Setup (20 perc)

Készíts Docker konfigurációt:

Dockerfile

- Multi-stage build használata
- Optimalizált Go binary
- Non-root user használata

docker-compose.yml

- Go aplikáció service
- PostgreSQL adatbázis service
- Volume mapping adatmegőrzéshez
- Environment variables

6. Adatbázis Migráció (10 perc)

- Automatikus adatbázis migráció alkalmazás indításakor
- Seed adatok betöltése (5-10 könyv)

7. Error Handling & Validation (10 perc)

- Proper HTTP status codes
- Input validation
- Database error handling
- JSON error responses

8. Dokumentáció (5 perc)

- README.md alapvető futtatási instrukciókat
- API endpoint dokumentáció

Technikai Specifikációk

Kötelező Packages

```
require (
    github.com/gin-gonic/gin v1.9.1
    gorm.io/gorm v1.25.4
    gorm.io/driver/postgres v1.5.2
    github.com/joho/godotenv v1.4.0
)
```

Environment Variables

```
DB_HOST=localhost
DB_PORT=5432
DB_USER=library_user
DB_PASSWORD=library_pass
DB_NAME=library_db
DB_SSLMODE=disable
PORT=8080
```

Projekt Struktúra (Javasolt)

```
library-api/
├── cmd/
│   └── main.go
├── internal/
│   ├── handlers/
│   │   ├── books.go
│   │   └── borrowings.go
│   ├── models/
│   │   ├── book.go
│   │   └── borrowing.go
│   ├── database/
│   │   ├── connection.go
│   │   └── migrate.go
│   └── middleware/
│       └── cors.go
├── docker-compose.yml
├── Dockerfile
├── .env
├── .gitignore
├── go.mod
├── go.sum
└── README.md
```

Értékelési Szempontok

Alapfunkciók (60 pont)

- [ ] CRUD műveletek működnek (20 pont)
- [ ] GORM integráció helyesen implementált (15 pont)
- [ ] HTTP endpoint-ok megfelelően válaszolnak (15 pont)
- [ ] Request validation működik (10 pont)

Kód Minőség (25 pont)

- [ ] Clean code elvek betartása (10 pont)
- [ ] Proper error handling (8 pont)
- [ ] Megfelelő projekt struktúra (7 pont)

Docker & DevOps (15 pont)

- [ ] Dockerfile optimalizált (8 pont)
- [ ] docker-compose.yml helyes konfiguráció (7 pont)

Bonus Feladatok (Ha maradt idő)

1. Egyszerű JWT Auth (20 perc)

- Basic token alapú authentikáció
- Protected endpoints

2. Logging (10 perc)

- Structured logging implementálása
- Request/response logging

3. Health Check Endpoint (5 perc)

- `GET /health` endpoint
- Database connection check

4. Swagger Dokumentáció (15 perc)

- API dokumentáció generálása
- Swagger UI integration

Elvárt Deliverables

1. **Működő Go alkalmazás** ami Docker-ben fut
2. **Teljes CRUD funkcionalitás** könyvekhez
3. **Kölcsönzési rendszer** alapfunkciók
4. **Docker setup** PostgreSQL-lel
5. **README.md** futtatási instrukciókat
6. **Postman Collection** vagy curl példák API teszteléshez

Sikeres Végrehajtás Lépései

1. Gyors Start (10 perc)

```
# Projekt inicializálás
mkdir library-api && cd library-api
go mod init library-api
go get github.com/gin-gonic/gin
go get gorm.io/gorm
go get gorm.io/driver/postgres
```

2. Minimális Működő Verzió (60 perc)

- Basic HTTP szerver
- Egy endpoint (GET /books)
- Adatbázis kapcsolat
- Docker alapbeállítás

3. Funkcionalitás Bővítés (40 perc)

- Összes CRUD endpoint
- Kölcsönzési logika
- Error handling
- Validáció

4. Finalizálás (10 perc)

- Dokumentáció
- Tesztelés
- Code cleanup

Értékelés Kritériumai

Kiváló (90-100%)

- Minden alap requirement teljesítve
- Clean, maintainable code
- Proper error handling
- Docker optimalizálva
- Bonus feladatok egy része megoldva

Jó (70-89%)

- Alap CRUD működik
- GORM helyesen használva
- Docker működik
- Kisebb hibák/hiányosságok

Elfogadható (50-69%)

- Részleges funkcionalitás
- Alapvető REST API működik
- Adatbázis kapcsolat OK
- Jelentős hiányosságok

Nem megfelelő (<50%)

- Nem működő alkalmazás
- Alapvető hibák

- Incomplete implementation

Tippek a Jelöltnek

1. **Időbeosztás**: Kezdj egy működő minimális verzióval
2. **Git**: Használj git-et, commitolj gyakran
3. **Tesztelés**: Minden endpoint-ot tesztelj le
4. **Dokumentáció**: Írj le hogyan kell futtatni
5. **Debugging**: Használj logolást problémák esetén

Sok sikert a feladat megoldásához! 🚀