Листинг 1 – Исходный текст модуля App

```
package app

import (
	"fmt"
	"log"

	"adboard/internal/config"
	"adboard/internal/handlers"
	"adboard/internal/middleware"
	"adboard/internal/models"
	"adboard/internal/repository"
	"adboard/internal/service"

	"github.com/labstack/echo/v4"
	echomiddleware "github.com/labstack/echo/v4/middleware"
	"gorm.io/driver/postgres"
	"gorm.io/gorm"
)

type App struct {
	cfg  *config.Config
	echo *echo.Echo
	db   *gorm.DB
}

func NewApp(cfg *config.Config) *App {
	return &App{cfg: cfg}
}

func (a *App) InitDB() error {
	log.Printf("Attempting to connect to database with config:
host=%s port=%d user=%s dbname=%s",
		a.cfg.DBHost, a.cfg.DBPort, a.cfg.DBUser,
a.cfg.DBName)
```

Продолжение листинга 1

```
	dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s
dbname=%s sslmode=disable",
		a.cfg.DBHost, a.cfg.DBPort, a.cfg.DBUser,
a.cfg.DBPassword, a.cfg.DBName)

	db, err := gorm.Open(postgres.Open(dsn), &gorm.Config{})
	if err != nil {
		log.Printf("Failed to connect to database: %v", err)
		return fmt.Errorf("failed to connect to
database: %w", err)
	}
	a.db = db
	log.Println("Successfully connected to database")
	return nil
}

func (a *App) InitializeData() error {
	// Только проверяем наличие категорий, миграции
выполняются отдельно
	err := a.checkCategoriesExist()
	if err != nil {
		return fmt.Errorf("failed to check categories: %w",
err)
	}
	return nil
}

func (a *App) InitEcho() {
	a.echo = echo.New()
	a.echo.Use(echomiddleware.Logger())
	a.echo.Use(echomiddleware.Recover())
	a.echo.Use(echomiddleware.CORS())
```

Продолжение листинга 1

```
	a.echo.Static("/images", a.cfg.UploadDir)
	a.SetupValidator()
}

func (a *App) RegisterRoutes() {
	authRepo := repository.NewAuthRepository(a.db)
	userRepo := repository.NewUserRepository(a.db)
	categoryRepo := repository.NewCategoryRepository(a.db)
	adRepo := repository.NewAdRepository(a.db)
	imageRepo := repository.NewImageRepository(a.db)

	authService := service.NewAuthService(authRepo, a.cfg)
	userService := service.NewUserService(userRepo, authRepo)
	categoryService :=
service.NewCategoryService(categoryRepo)
	adService := service.NewAdService(adRepo)
	imageService := service.NewImageService(imageRepo, adRepo)

	authHandler := handlers.NewAuthHandler(authService)
	userHandler := handlers.NewUserHandler(userService)
	categoryHandler :=
handlers.NewCategoryHandler(categoryService)
	adHandler := handlers.NewAdHandler(adService)
	imageHandler := handlers.NewImageHandler(imageService)

	a.echo.POST("/register", authHandler.Register)
	a.echo.POST("/login", authHandler.Login)
	a.echo.GET("/categories", categoryHandler.GetCategories)
	a.echo.GET("/ads", adHandler.SearchAds)
	a.echo.GET("/ads/:id", adHandler.GetAd)

	protected := a.echo.Group("")
	protected.Use(middleware.JWTAuth(a.cfg))
```

Продолжение листинга 1

```
	protected.GET("/profile", userHandler.GetProfile)
	protected.PUT("/profile", userHandler.UpdateProfile)

	protected.POST("/ads", adHandler.CreateAd)
	protected.POST("/ads/with-images",
adHandler.CreateAdWithImages, middleware.UploadFiles)
	protected.GET("/my-ads", adHandler.GetUserAds)
	protected.PUT("/ads/:id", adHandler.UpdateAd)
	protected.DELETE("/ads/:id", adHandler.DeleteAd)

	protected.POST("/ads/:id/images", imageHandler.AddImages,
middleware.UploadFiles)
	protected.DELETE("/images/:imageId",
imageHandler.DeleteImage)
}

func (a *App) Start() error {
	addr := fmt.Sprintf(":%s", a.cfg.ServerPort)
	log.Printf("Server starting on %s", addr)
	return a.echo.Start(addr)
}

func (a *App) checkCategoriesExist() error {
	// Проверяем, есть ли уже категории
	var count int64
	err := a.db.Model(&models.Category{}).Count(&count).Error
	if err != nil {
		return err
	}

	// Если категорий нет, то предупреждаем (не ошибка, так
как это может быть первый запуск)
```

Продолжение листинга 1

```
	if count == 0 {
		log.Println("Внимание: Категории не найдены.
Убедитесь, что выполнены все миграции: make db-up")
	}

	return nil
}
```

Ссылка на местоположение полного комплекта исходных модулей программы – https://github.com/cod1ng-space/avito-clone