Golang Final Project

Team members:

- Dauren Maskeugaliyev (210103184)
- Azamat Kozhakov (210107116)
- Zhambyl Dauletkhan (210107089)

Overview:

Educational platform with teachers, administrators and students. Platform allows teachers to create courses and make plans for them with adding materials. Students can subscribe for courses and get notifications whenever changes are made in the courses they followed. Analog of stepik.org

MVP:

MVP contains 7 main features:

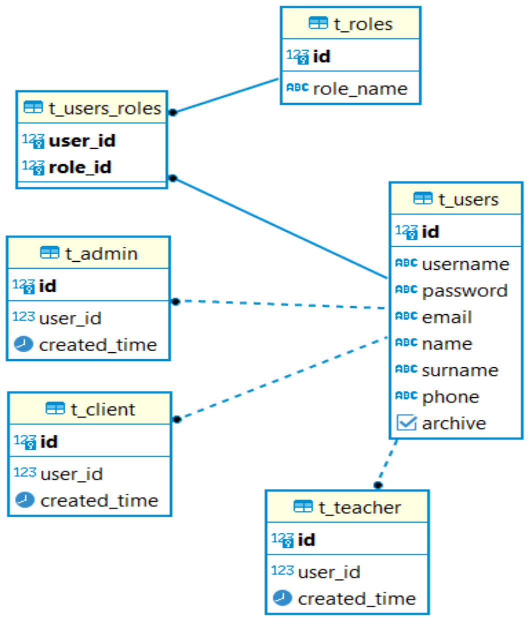
1. Registration of client
2. Authorization of user
3. Creating courses
4. Creating plan with validation of course owner
5. Register to courses
6. Notification any updates of course
7. Course catalog with pagination

Architecture Design:

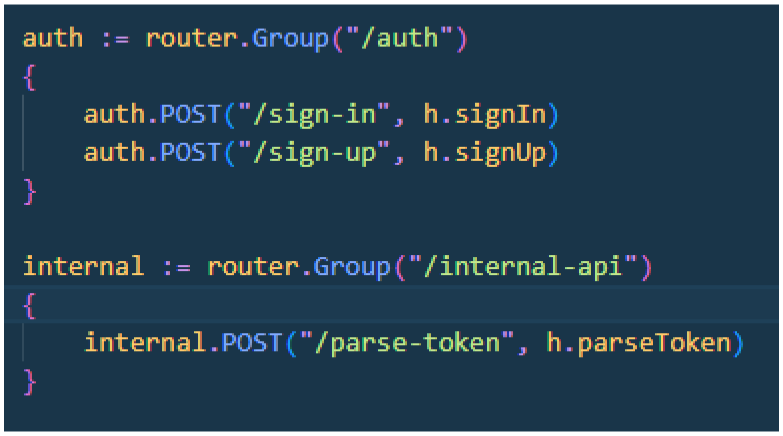
We used microservices architecture with 3 separated services: AuthService, CourseManagementService, NotificationService.

AuthService (Link):

- Purpose: Students registration and user authorization



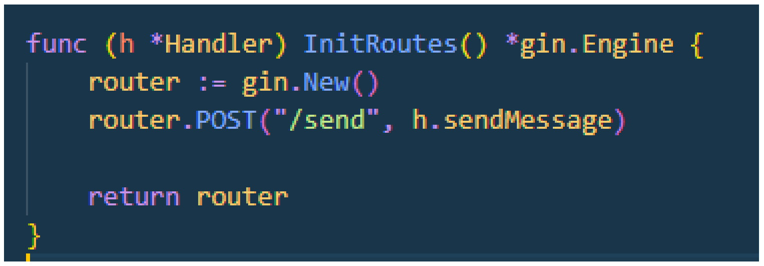
- Scheme:
- Endpoints:



```
auth := router.Group("/auth")
{
    auth.POST("/sign-in", h.signIn)
    auth.POST("/sign-up", h.signUp)
}

internal := router.Group("/internal-api")
{
    internal.POST("/parse-token", h.parseToken)
}
```

NotificationService (Link):

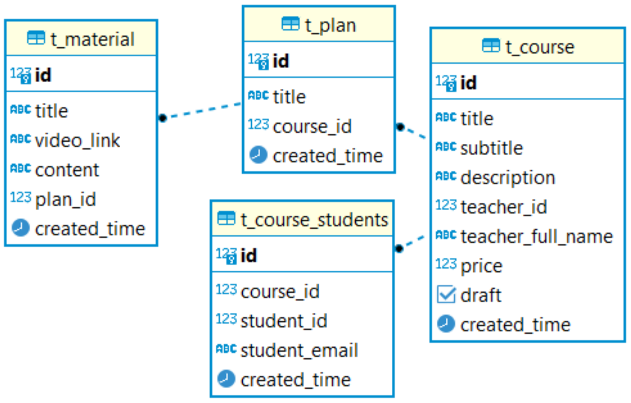
- Purpose: Mailing notifications about changes in course
- Endpoints:



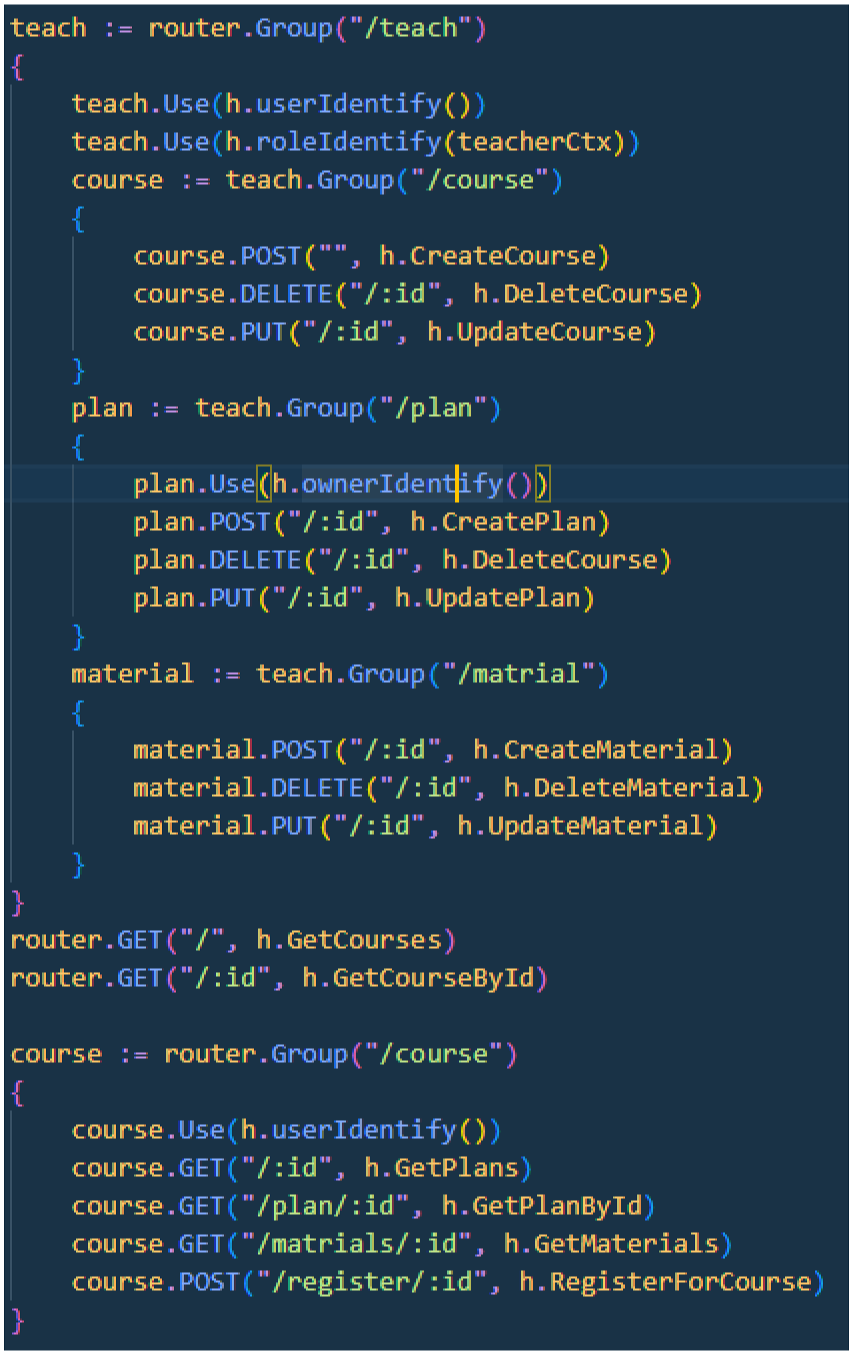
```
func (h *Handler) InitRoutes() *gin.Engine {
    router := gin.New()
    router.POST("/send", h.sendMessage)

    return router
}
```

CourseManagementService (Link):

- Purpose: Core of project with ability to manage courses


t_material
id
title
video_link
content
plan_id
created_time
t_plan
id
title
course_id
created_time
t_course
id
title
subtitle
description
teacher_id
teacher_full_name
price
draft
created_time
t_course_students
id
course_id
student_id
student_email
created_time

- Schema:

* Endpoints: 

```
teach := router.Group("/teach")
{
    teach.Use(h.userIdentify())
    teach.Use(h.roleIdentify(teacherCtx))
    course := teach.Group("/course")
    {
        course.POST("", h.CreateCourse)
        course.DELETE("/:id", h.DeleteCourse)
        course.PUT("/:id", h.UpdateCourse)
    }
    plan := teach.Group("/plan")
    {
        plan.Use(h.ownerIdentify())
        plan.POST("/:id", h.CreatePlan)
        plan.DELETE("/:id", h.DeleteCourse)
        plan.PUT("/:id", h.UpdatePlan)
    }
    material := teach.Group("/matrial")
    {
        material.POST("/:id", h.CreateMaterial)
        material.DELETE("/:id", h.DeleteMaterial)
        material.PUT("/:id", h.UpdateMaterial)
    }
}
router.GET("/", h.GetCourses)
router.GET("/:id", h.GetCourseById)

course := router.Group("/course")
{
    course.Use(h.userIdentify())
    course.GET("/:id", h.GetPlans)
    course.GET("/plan/:id", h.GetPlanById)
    course.GET("/matrials/:id", h.GetMaterials)
    course.POST("/register/:id", h.RegisterForCourse)
}
```

Setup Instructions

1. Git clone https://github.com/senitapqan/AuthService.git
2. Git clone https://github.com/senitapqan/CourseService.git
3. Git clone https://github.com/senitapqan/NotificationService.git
4. Go mod download in every microservice
5. Setup configuration in every microservice
6. Go run cmd/main.go (AuthService)

7. Go run cmd/main.go (NotificationService)
8. Go run cmd/main.go (CourseService)

Summary:

This documentation file provides an overview of each microservice, installation steps, running instructions, API endpoints