



# Разработка приложения №1

“Программирование сетевых приложений”

Лекция 4



# Работа с Git



# Работа с Git

## Установка

- |            |                                       |
|------------|---------------------------------------|
| 1) Linux   | <code>sudo apt-get install git</code> |
| 2) macOS   | <code>brew install git</code>         |
| 3) Windows | “Git for windows”                     |



<https://git-scm.com/download/win>



# Работа с Git

## Создание нового репозитория



Переходим в папку с проектом



```
git init
```



```
git remote add origin https://github.com/dozen/na.git
```



```
git push -u origin master
```



# Работа с Git

## Клонирование существующего репозитория



Переходим в папку для будущего проекта



```
git clone https://github.com/dozen/na.git na
```



# Работа с Git

Работа с репозиторием: Подготовка файлов



Делаем какие-то изменения



```
git add main.go
```

Один файлов



```
git add -A
```

Все файлы



```
git add .
```

Без удалений

Добавляем изменённые файлы



# Работа с Git

Работа с репозиторием: Коммит



Подготовили файлы



```
git commit -m 'Add main file'
```

Фиксируем изменение



# Работа с Git

Работа с репозиторием



Сделали коммит



```
git push
```

Отправляем изменения





# Работа с Git

Работа с репозиторием



На сервере произошли изменения



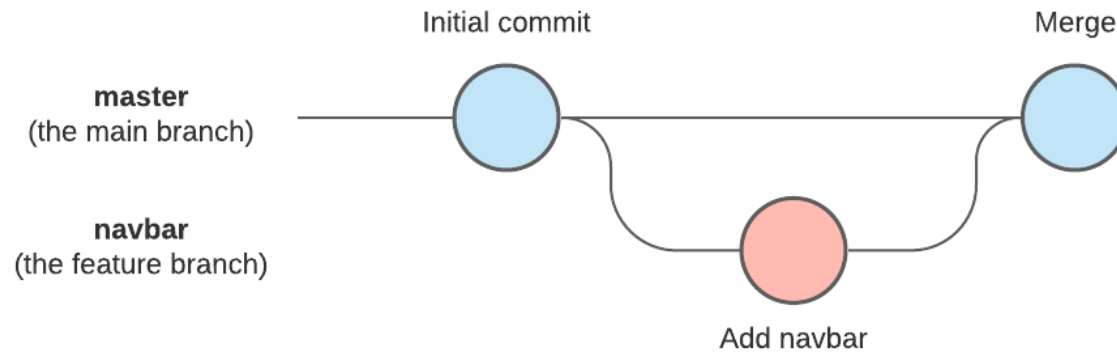
```
git pull
```

Стягиваем изменения



# Работа с Git

## Ветки



>\_

```
git branch navbar
```

>\_

```
git checkout navbar
```

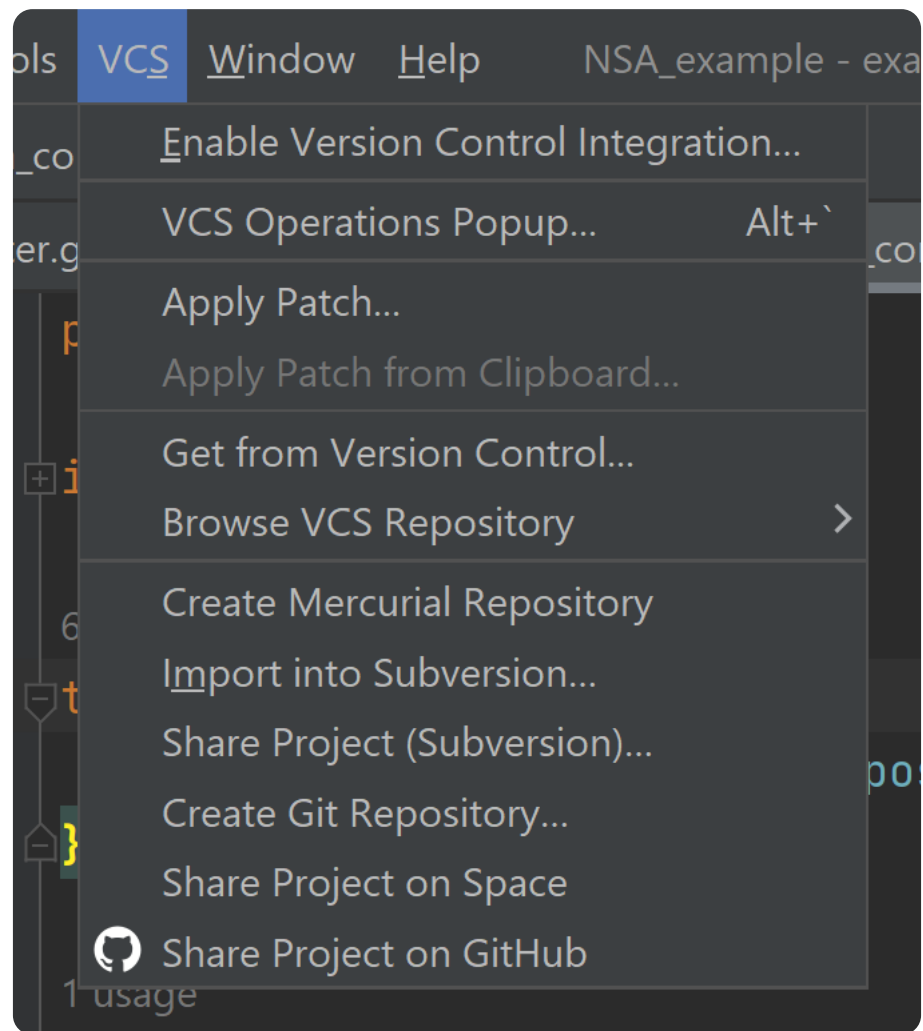
>\_

```
git merge navbar
```



# Работа с Git


GUI





# Работа с Git

## GUI

 Share Project On GitHub ✕

Repository name:

Exam\_process\_web\_application\_MEPHI\_NA

☐ Private

Remote:

origin

Description:

Пример веб-приложения для курса ПСП МИФИ

Share by:

github.com/grigorevmp

[Add account](#)

?

Share

Cancel

?

Share

Cancel

Share by:

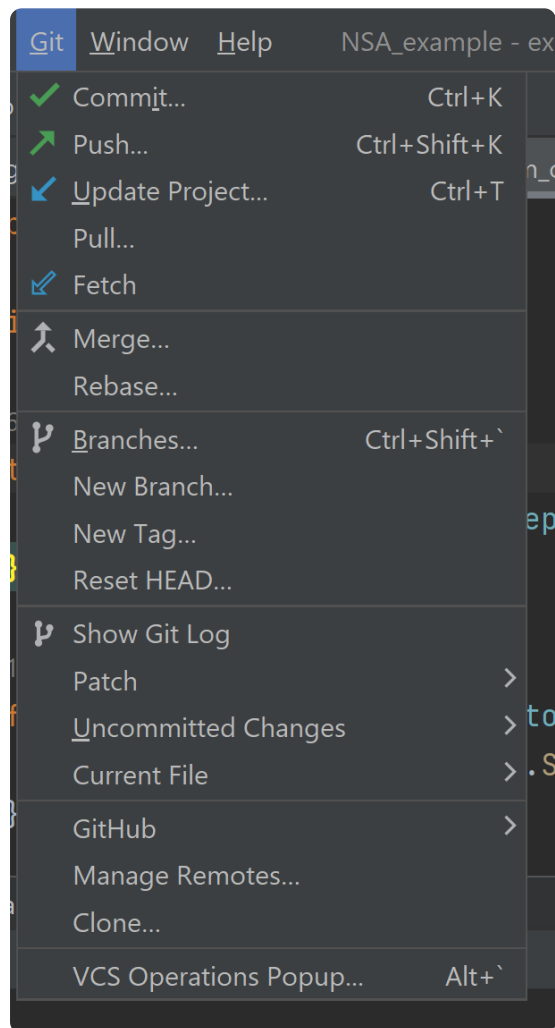
github.com/grigorevmp

[Add account](#)



# Работа с Git

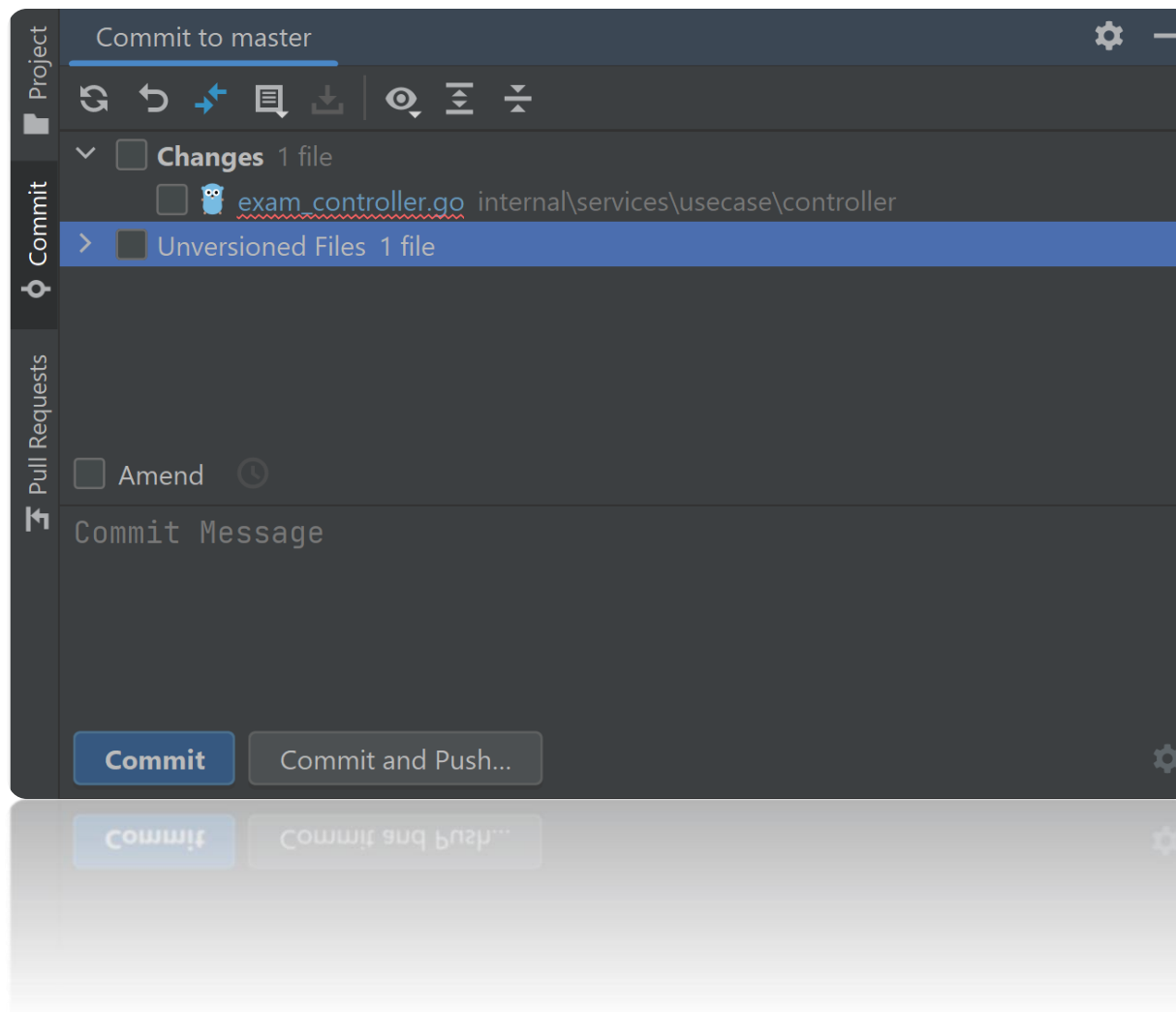
## GUI





# Работа с Git

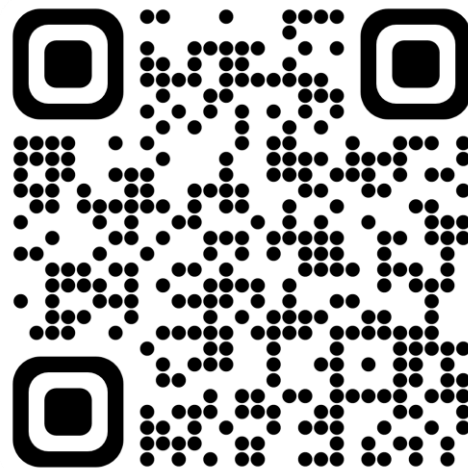
## GUI





# Работа с Git

Подробнее



<https://proglib.io/p/git-for-half-an-hour>



# Постановка задачи





# Постановка задачи

Разработать приложение автоматизирующее процесс проведения экзамена по программированию на кафедре №12.

- 1) Возможность управления экзаменом: список участников и задач
- 2) Возможность фиксировать сдачу экзамена студентом
- 3) Возможность просмотра информации об экзамене



# Проектирование БД



# Обсуждение

- 1) Возможность управления экзаменом: список участников и задач
- 2) Возможность фиксировать сдачу экзамена студентом
- 3) Возможность просмотра информации об экзамене



# Обсуждение

- 1) Возможность управления **экзаменом**: список **участников** и **задач**
- 2) Возможность фиксировать сдачу экзамена студентом
- 3) Возможность просмотра информации об экзамене



# Обсуждение

- 1) Возможность управления экзаменом: список участников и задач
- 2) Возможность фиксировать сдачу экзамена студентом
- 3) Возможность просмотра информации об экзамене



# Обсуждение

- 1) Возможность **управления** экзаменом: список участников и задач
- 2) Возможность фиксировать **сдачу экзамена** студентом
- 3) Возможность **просмотра информации** об экзамене



# Обсуждение

экзамен

участник

задача



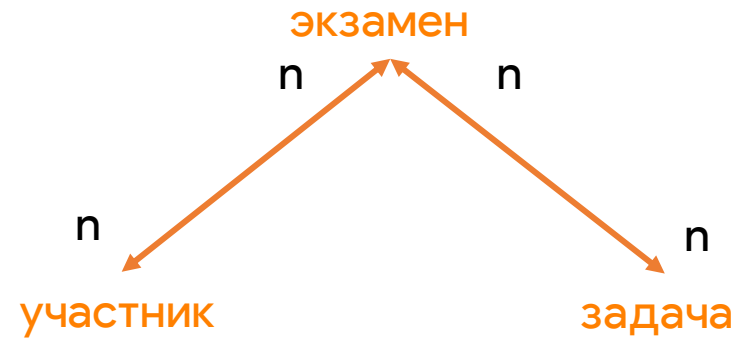
# Обсуждение







# Обсуждение





# Проектирование БД

## Экзаменационный лист

### ExamCard

ID	int
Variant	int
Name	string
Task	int []
Date	time.Time



# Проектирование БД

Пользователь

User

ID	int
Name	string
Group	string



# Проектирование БД

Задача

Task

ID	int
Name	string
Desc	string
MaxScore	int



# Проектирование БД

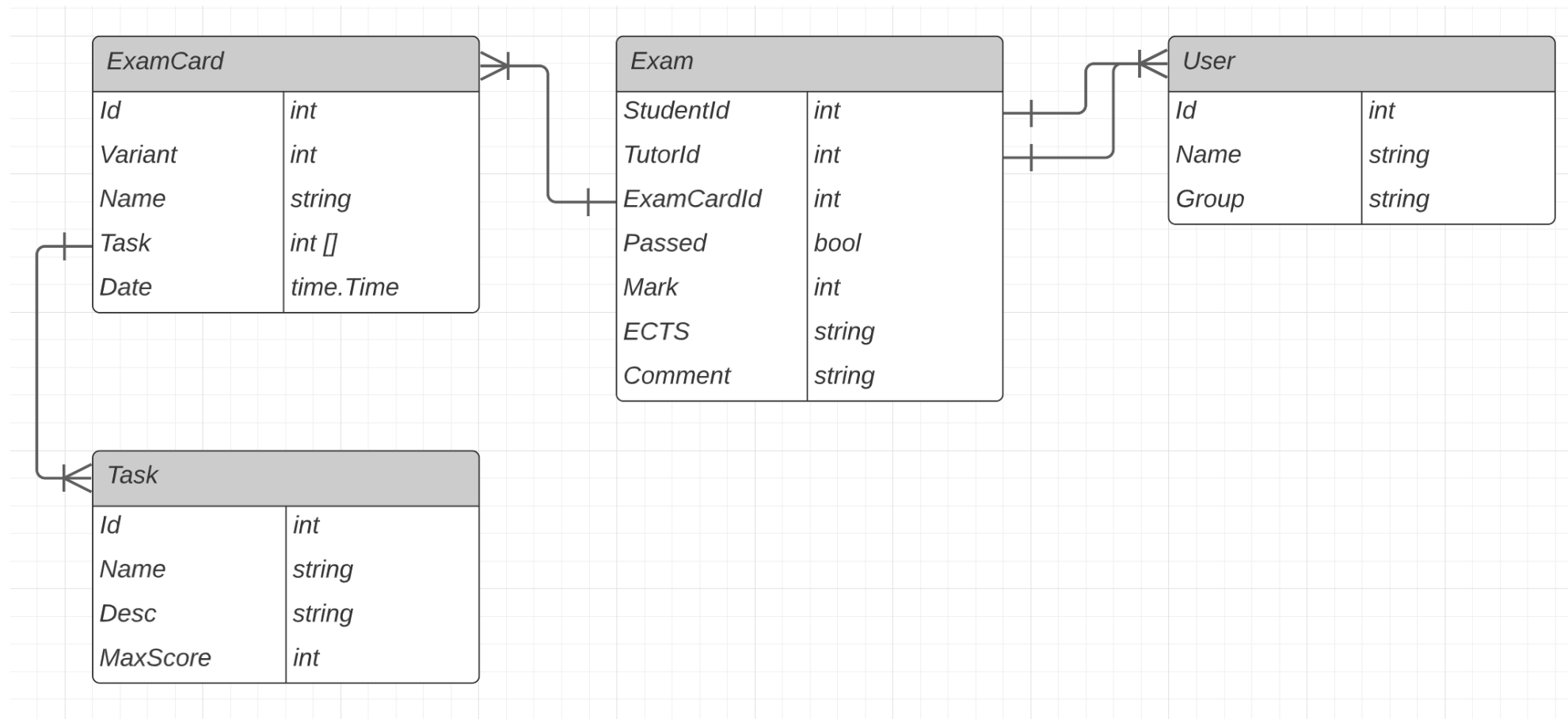
## Диаграмма





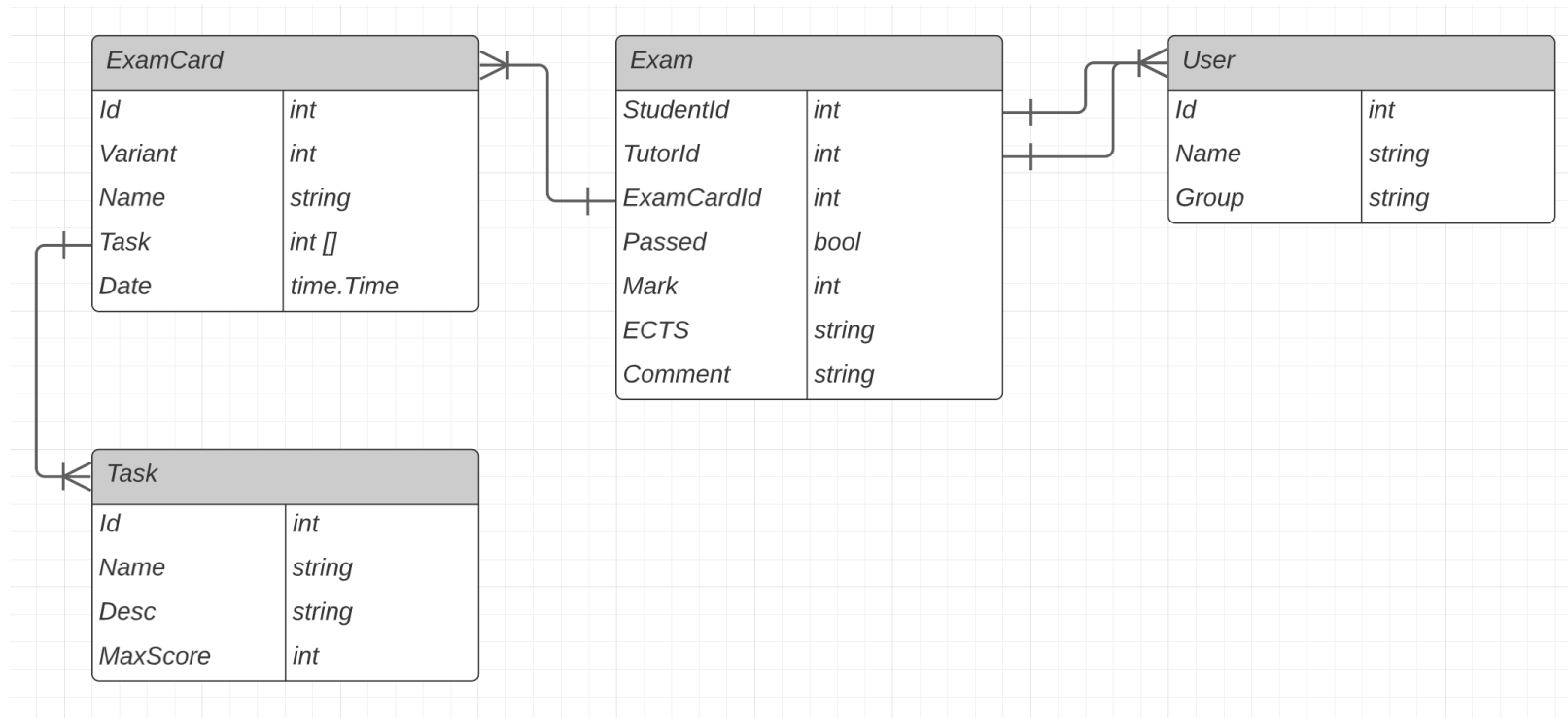
# Проектирование БД

## Диаграмма





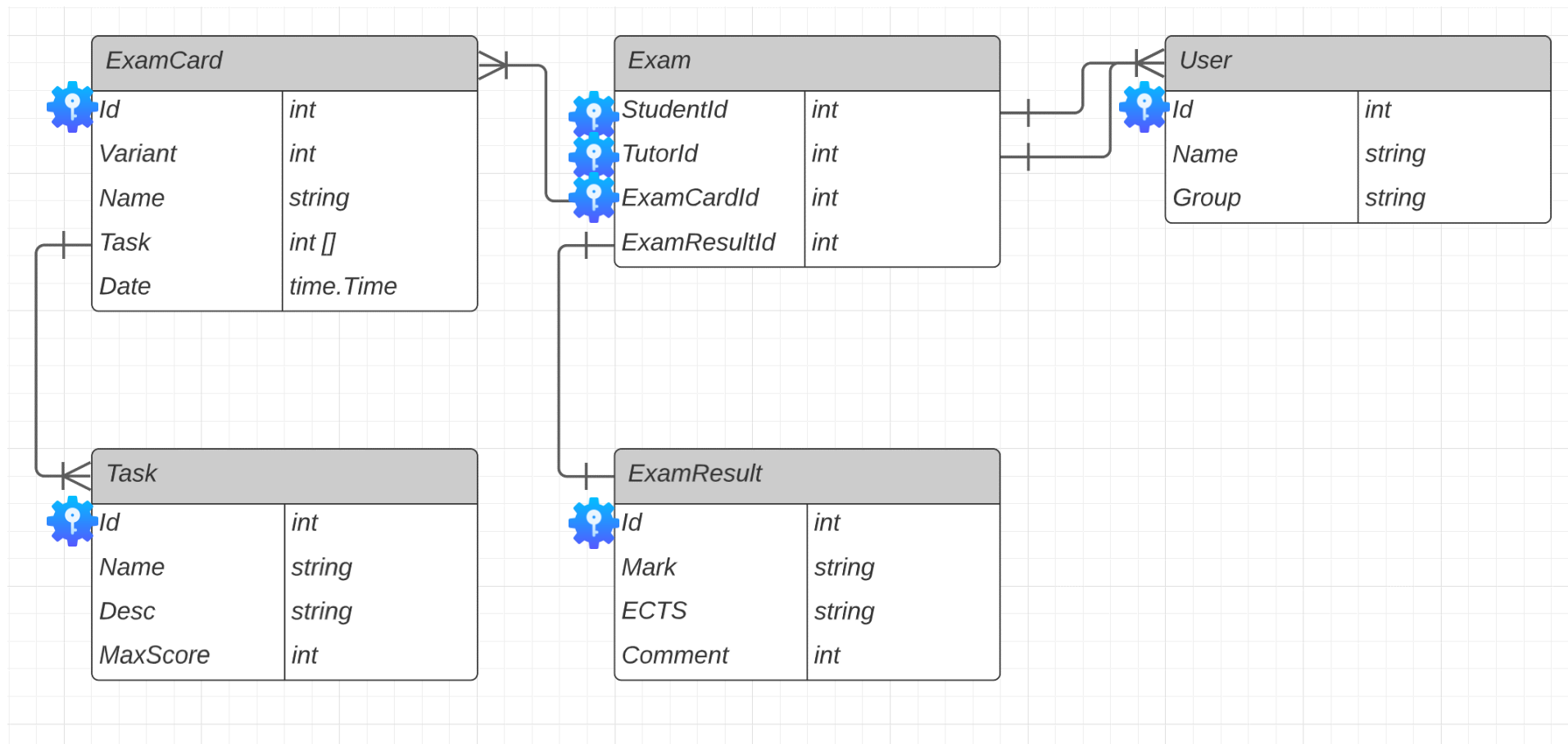
# Обсуждение





# Проектирование БД

## Диаграмма

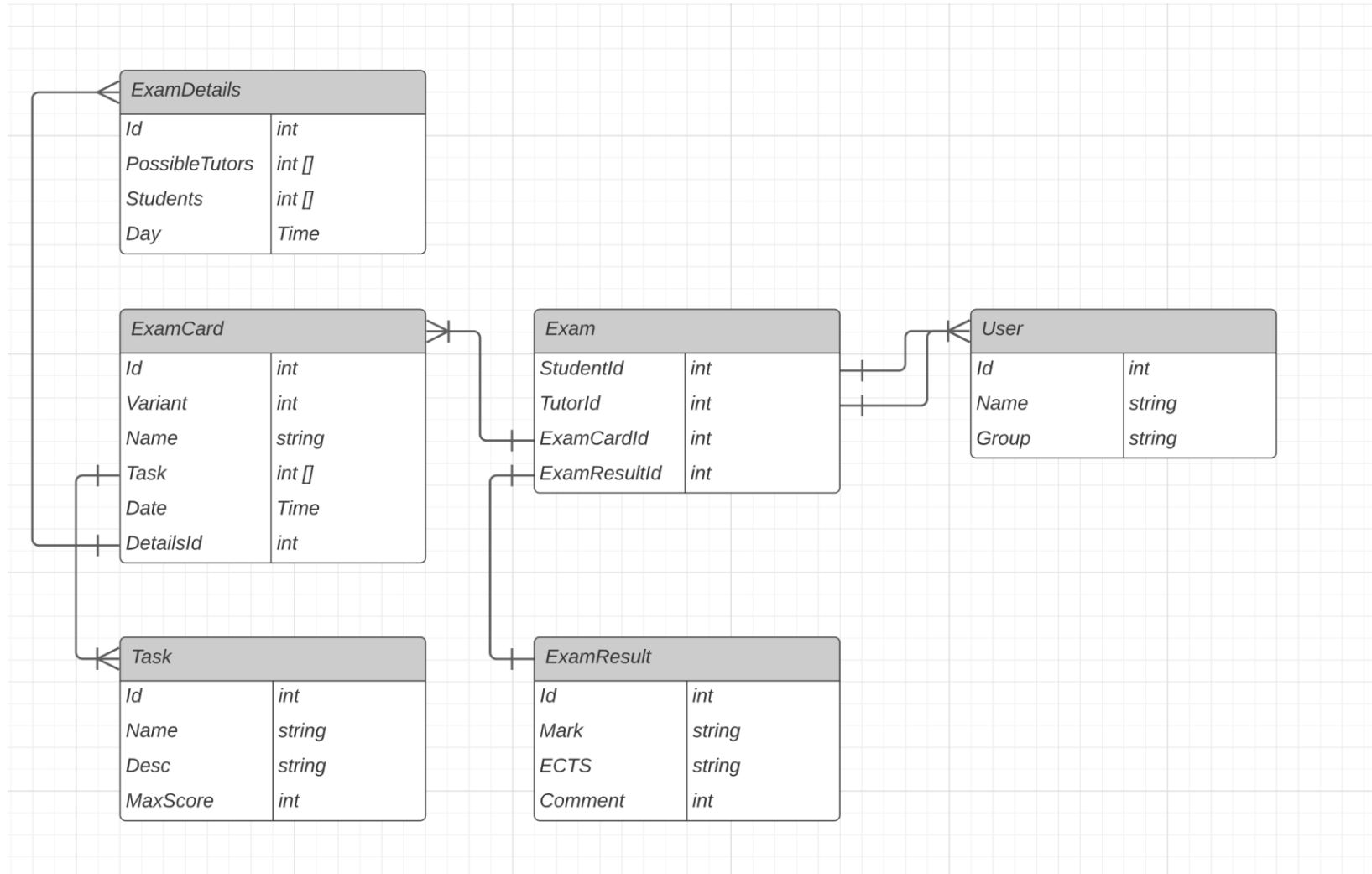






# Проектирование БД

## Диаграмма





# Установка pgAdmin



# pgAdmin

Установка



<https://www.postgresql.org/download/>

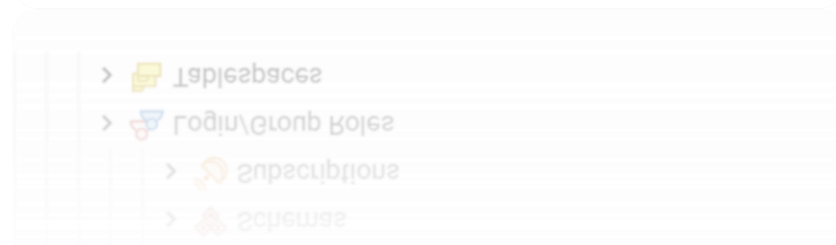
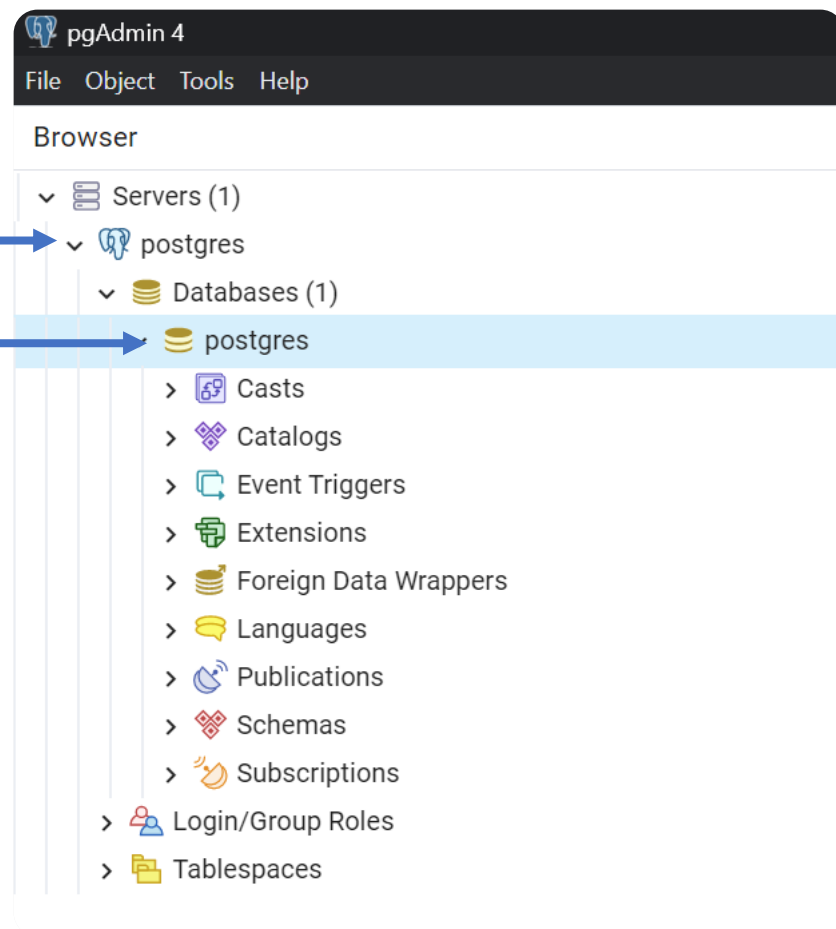


# pgAdmin

## Настройка

Сервер

БД





# pgAdmin

## properties

postgres

General

Connection

Parameters

SSH Tunnel

Advanced

Host name/address

localhost

Port

5432

Maintenance database

postgres

Username

postgres

Kerberos authentication?

☐

Role

Service

i

?

Close

Reset

Save

!

i

Close

Reset

Save



# pgAdmin

## Таблицы

- ▼ Schemas (1)
  - ▼ public
    - > Aggregates
    - > Collations
    - > Domains
    - > FTS Configurations
    - > FTS Dictionaries
    - > FTS Parsers
    - > FTS Templates
    - > Foreign Tables
    - > Functions
    - > Materialized Views
    - > Operators
    - > Procedures
    - > 1..3 Sequences
    - ▼ Tables (4)
      - > exam\_cards
      - > exams
      - > user
      - > users
    - > Trigger Functions
    - > Types
    - > Views



# Структура проекта



# Структура проекта

/cmd

/internal

    /app

    /config

    /database

        (слой базы данных)

    /models

        (слой базы данных)

    /services

        (бизнес-слой)

    /transport

        (транспортный слой)

/build

/docs

README.md







<Кooooooooooooоод/>



# Сущности БД

examCard.go

```
package models

import "time"

// ExamCard
// Describes examination card
type ExamCard struct {
    Id      int    `json:"id" gorm:"primary_key"`
    Variant int    `json:"variant_number"`
    Name    string `json:"name"`
    Task    string `json:"task"`
    Date    time.Time `json:"date"`
}
```

/cmd  
/internal

/app  
/config  
/database  
**/models**  
/services  
/transport

/build  
/docs  
README.md



# Сущности БД

exam.go

/cmd  
/internal

/app  
/config  
/database  
/models  
/services  
/transport

/build  
/docs  
README.md

```
package models

// Exam
// Describes user exam
type Exam struct {
    StudentId    int    `json:"student" gorm:"primary_key"`
    TutorId      int    `json:"tutor" gorm:"primary_key"`
    ExamCardId   int    `json:"exam_card" gorm:"primary_key"`
    Passed       bool   `json:"passed"`
    Mark         int    `json:"mark"`
    ECTS         string `json:"ects"`
    Comment      string `json:"comment"`
    User         User   `json:"user" gorm:"foreignKey:StudentId"`
    Tutor        User   `json:"tutor" gorm:"foreignKey:TutorId"`
    ExamCard     ExamCard `json:"examCard" gorm:"foreignKey:ExamCardId"`
}
```



# Сущности БД

user.go

```
package models

// User
// Describes Dozen user: Student or Tutor
type User struct {
    Id      int      `json:"id" gorm:"primary_key"`
    Name    string   `json:"name"`
    Group   string   `json:"group"`
}
```

/cmd  
/internal

/app  
/config  
/database  
**/models**  
/services  
/transport

/build  
/docs  
README.md



# Создание БД

Interfaces/SqlHandler.go

```
package interfaces

type SqlHandler interface {
    Create(object interface{})
    FindAll(object interface{})
    Where(object interface{}, conds ...interface{}) (tx *gorm.DB)
    Preload(query string, args ...interface{}) (tx *gorm.DB)
    DeleteById(object interface{}, id string)
}
```

/cmd  
/internal

/app  
/config  
/database  
/models  
/services  
/transport

/build  
/docs  
README.md



# Создание БД

... in goLang

```
package main

import "fmt"

func main() {
    sayHello()
    sayHello("Rahul")
    sayHello("Mohit", "Rahul", "Rohit", "Johny")
}

// using Ellipsis
func sayHello(names ...string) {
    for _, n := range names {
        fmt.Printf("Hello %s\n", n)
    }
}
```

```
}

}

    fmt.Printf("Hello %s\n", n)
    for _, n := range names {
```



# Создание БД

```
Where(object interface{}, conds ...interface{}) (tx *gorm.DB)
```

```
gorm.DB.Where(...).Where(...) .Where(...)
```



# Создание БД

## SqlHandler.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
package database

import (
    "NSA_example/internal/database/interfaces"
    "gorm.io/driver/postgres"
    "gorm.io/gorm"
)

type SqlHandler struct {
    db *gorm.DB
}

func NewSqlHandler() interfaces.SqlHandler {
    dbServer := "host=localhost user=postgres password=admin dbname=postgres port=5432 sslmode=disable"
    db, err := gorm.Open(postgres.Open(dbServer), &gorm.Config{})

    if err != nil {
        panic(err.Error)
    }
    sqlHandler := new(SqlHandler)
    sqlHandler.db = db
    return sqlHandler
}

func (handler *SqlHandler) Create(obj interface{}) {
    handler.db.Create(obj)
}

func (handler *SqlHandler) FindAll(obj interface{}) {
    handler.db.Find(obj)
}

func (handler *SqlHandler) DeleteById(obj interface{}, id string) {
    handler.db.Delete(obj, id)
}
```





# Создание БД

SqlHandler.go

```
host=localhost  
user=postgres  
password=admin  
dbname=postgres  
port=5432  
sslmode=disable
```

The screenshot shows a window titled "postgres" with a tabbed interface. The "Connection" tab is selected. The form contains the following fields and controls:

- Host name/address: localhost
- Port: 5432
- Maintenance database: postgres
- Username: postgres
- Kerberos authentication?: ☐
- Role: (empty field)
- Service: (empty field)

At the bottom of the window, there are buttons for "Close", "Reset", and "Save".



# Создание БД

SqlHandler.go

```
func (handler *SqlHandler) Where(object interface{}, args ...interface{}) (tx *gorm.DB) {  
    return handler.db.Where(object, args)  
}  
  
func (handler *SqlHandler) Preload(query string, args ...interface{}) (tx *gorm.DB) {  
    return handler.db.Preload(query, args)  
}
```

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md



# Создание БД

user\_repository.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
package database

import (
    "NSA_example/internal/database/interfaces"
    "NSA_example/internal/models"
)

type UserRepository struct {
    interfaces.SqlHandler
}

func (db *UserRepository) Store(u models.User) {
    db.Create(&u)
}

func (db *UserRepository) Select() []models.User {
    var user []models.User
    db.FindAll(&user)
    return user
}

func (db *UserRepository) Delete(id string) {
    var user []models.User
    db.DeleteById(&user, id)
}
```



# Создание БД

exam\_repository.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
package database

import (
    "NSA_example/internal/database/interfaces"
    "NSA_example/internal/models"
)

type ExamRepository struct {
    interfaces.SqlHandler
}

func (db *ExamRepository) Store(e models.Exam) {
    db.Create(&e)
}

func (db * ExamRepository) Select() []models.Exam {
    var exam []models.Exam
    db.FindAll(&exam)
    return exam
}

func (db * ExamRepository) Delete(id string) {
    var exam []models.Exam
    db.DeleteById(&exam, id)
}
```



# Создание БД

exam\_repository.go

```
func (db *ExamRepository) SelectUserResult(userId int, examCardId int) []models.Exam {  
    var exam []models.Exam  
    db.Preload("User").Preload("ExamCard").Preload("Tutor").Where("student_id = ?", userId).Where("exam_card_id = ?", examCardId).Find(&exam)  
    return exam  
}
```

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md



# Создание БД

exam\_card\_repository.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
package database

import (
    "NSA_example/internal/database/interfaces"
    "NSA_example/internal/models"
)

type ExamCardRepository struct {
    interfaces.SqlHandler
}

func (db *ExamCardRepository) Store(ec models.ExamCard) {
    db.Create(&e)
}

func (db * ExamCardRepository) Select() []models.ExamCard {
    var examCard []models.ExamCard
    db.FindAll(&examCard)
    return examCard
}

func (db * ExamCardRepository) Delete(id string) {
    var examCard []models.ExamCard
    db.DeleteById(&examCard, id)
}
```



# Бизнес логика

context.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
package services

type Context interface {
    Param(string) string
    Bind(interface{}) error
    Status(int)
    JSON(int, interface{})
}
```



# Бизнес логика

user\_repository.go

/cmd

/internal

/app

/config

/database

/models

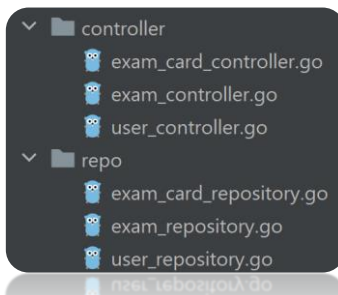
/services

/transport

/build

/docs

README.md



```
package repo

import "NSA_example/internal/models"

type UserRepository interface {
    Store(models.User)
    Select() []models.User
    Delete(id string)
}
```





# Бизнес логика

exam\_repository.go

/cmd

/internal

/app

/config

/database

/models

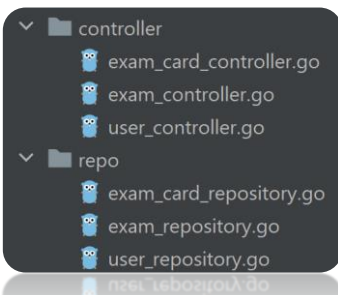
/services

/transport

/build

/docs

README.md



```
type ExamRepository interface {  
    Store(models.Exam)  
    Select() []models.Exam  
    SelectUserResult(userId int, examCardId int) []models.Exam  
    Delete(id string)  
}
```



# Бизнес логика

user\_controller.go

/cmd

/internal

/app

/config

/database

/models

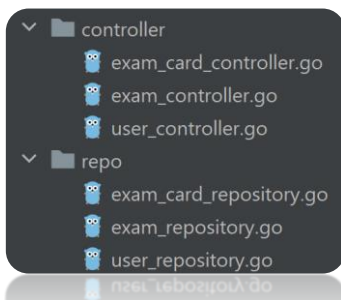
/services

/transport

/build

/docs

README.md



```
package controller

import (
    "NSA_example/internal/models"
    "NSA_example/internal/services/usecase/repo"
)

type UserInteractor struct {
    UserRepository repo.UserRepository
}

func (interactor *UserInteractor) Add(u models.User) {
    interactor.UserRepository.Store(u)
}

func (interactor *UserInteractor) GetInfo() []models.User {
    return interactor.UserRepository.Select()
}

func (interactor *UserInteractor) Delete(id string) {
    interactor.UserRepository.Delete(id)
}
```



# Бизнес логика

user\_controller.go

exam\_controller.go

exam\_card\_controller.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
package services

import ...

type UserController struct {
    Interactor controller.UserInteractor
}

func NewUserController(sqlHandler interfaces.SqlHandler) *UserController {
    return &UserController{
        Interactor: controller.UserInteractor{
            UserRepository: &database.UserRepository{
                SqlHandler: sqlHandler,
            },
        },
    }
}

func (controller *UserController) Create(c echo.Context) {
    u := models.User{}
    c.Bind(&u)
    controller.Interactor.Add(u)
    createdUsers := controller.Interactor.GetInfo()
    c.JSON(201, createdUsers)
    return
}

func (controller *UserController) GetUser() []models.User {
    res := controller.Interactor.GetInfo()
    return res
}

func (controller *UserController) Delete(id string) {
    controller.Interactor.Delete(id)
}
```



# Бизнес логика

user\_controller.go

exam\_controller.go

exam\_card\_controller.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
func (controller *ExamController) GetExamForUser(userId int, examCardId int) string {
    res := controller.Interactor.SelectUserResult(userId, examCardId)

    str := ""
    for _, exam := range res {
        str += "Студент " + exam.User.Name
            /\+ " сдал " + exam.Tutor.Name
            /\+ " на оценку " + strconv.Itoa(exam.Mark) + "(" + exam.ECTS + ")"
            /\+ ": " + exam.Comment
    }

    return str
}
```



# API

router.go

/cmd  
/internal

/app  
/config  
/database  
/models  
/services  
/transport

/build  
/docs  
README.md

```
package transport

import (
    "NSA_example/internal/database"
    controllers "NSA_example/internal/services"

    "github.com/labstack/echo"
    "net/http"
)

func Init() {
    // Echo instance
    e := echo.New()

    userController := controllers.NewUserController(database.NewSqlHandler())
    examCardController := controllers.NewExamCardController(database.NewSqlHandler())
    examController := controllers.NewExamController(database.NewSqlHandler())

    ...
}
```



# API

router.go

/cmd  
/internal

/app  
/config  
/database  
/models  
/services  
/transport

/build  
/docs  
README.md

```
...  
  
examController := controllers.NewExamController(database.NewSqlHandler())  
  
// Users  
  
e.GET("/users", func(c echo.Context) error {  
    users := userController.GetUser()  
    c.Bind(&users)  
    return c.JSON(http.StatusOK, users)  
})  
  
e.POST("/users", func(c echo.Context) error {  
    userController.Create(c)  
    return c.String(http.StatusOK, "created")  
})  
  
e.DELETE("/users/:id", func(c echo.Context) error {  
    id := c.Param("id")  
    userController.Delete(id)  
    return c.String(http.StatusOK, "deleted")  
})  
  
...
```



# API

router.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
// ExamCard
```

```
e.GET("/exam_cards", func(c echo.Context) error {  
    users := examCardController.GetExamCard()  
    c.Bind(&users)  
    return c.JSON(http.StatusOK, users)  
})
```

```
e.POST("/exam_cards", func(c echo.Context) error {  
    examCardController.Create(c)  
    return c.String(http.StatusOK, "created")  
})
```

```
e.DELETE("/exam_cards/:id", func(c echo.Context) error {  
    id := c.Param("id")  
    examCardController.Delete(id)  
    return c.String(http.StatusOK, "deleted")  
})
```



# API

router.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
// Exam
```

```
e.GET("/exams", func(c echo.Context) error {  
    users := examController.GetExam()  
    c.Bind(&users)  
    return c.JSON(http.StatusOK, users)  
})
```

```
e.POST("/pass_exam", func(c echo.Context) error {  
    examController.Create(c)  
    return c.String(http.StatusOK, "created")  
})
```

```
e.DELETE("/users/:id", func(c echo.Context) error {  
    id := c.Param("id")  
    examController.Delete(id)  
    return c.String(http.StatusOK, "deleted")  
})
```

```
}
```





# API

router.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
e.GET("/exam/:userId/:examId", func(c echo.Context) error {
    userId, _ := strconv.Atoi(c.Param("userId"))
    examId, _ := strconv.Atoi(c.Param("examId"))
    exam := examController.GetExamForUser(userId, examId)
    c.Bind(&exam)
    return c.JSON(http.StatusOK, exam)
})
```



# API

router.go

```
package config

const (
    ServerPort = ":1323"
    DatabaseUrl = "host=localhost user=postgres password=admin dbname=postgres port=5432 sslmode=disable"
)
```

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md



# Server

apiserver.go

/cmd

/internal

/app

/config

/database

/models

/services

/transport

/build

/docs

README.md

```
package app

import ...

type Server struct {
    state bool
}

func (server *Server) Start() {
    dbinit()
    transport.Init()
    e := echo.New()
    e.Logger.Fatal(e.Start(config.ServerPort))
}

func dbinit() {

    db, err := gorm.Open(postgres.Open(config.DatabaseUrl), &gorm.Config{})

    err = db.Migrator().CreateTable(models.User{})
    if err != nil {
        fmt.Print("User already exists")
    }
    err = db.Migrator().CreateTable(models.Exam{})
    if err != nil {
        fmt.Print("Exam already exists")
    }
    err = db.Migrator().CreateTable(models.ExamCard{})
    if err != nil {
        fmt.Print("ExamCard already exists")
    }
}
```



# Main

main.go

```
package main

import "NSA_example/internal/app"

func main() {
    var server = app.Server{}
    server.Start()
}
```

/cmd  
/internal  
/build  
/docs  
README.md



# Config

go.mod

```
module NSA_example  
  
go 1.20
```

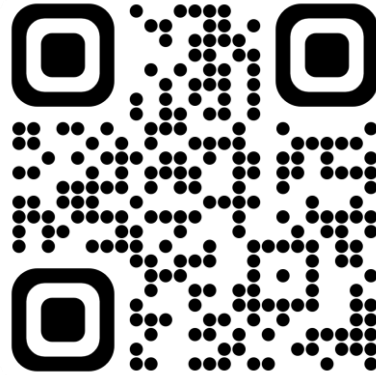
/cmd  
/internal  
/



# Тестирование



# Тестирование



<https://web.postman.co/>

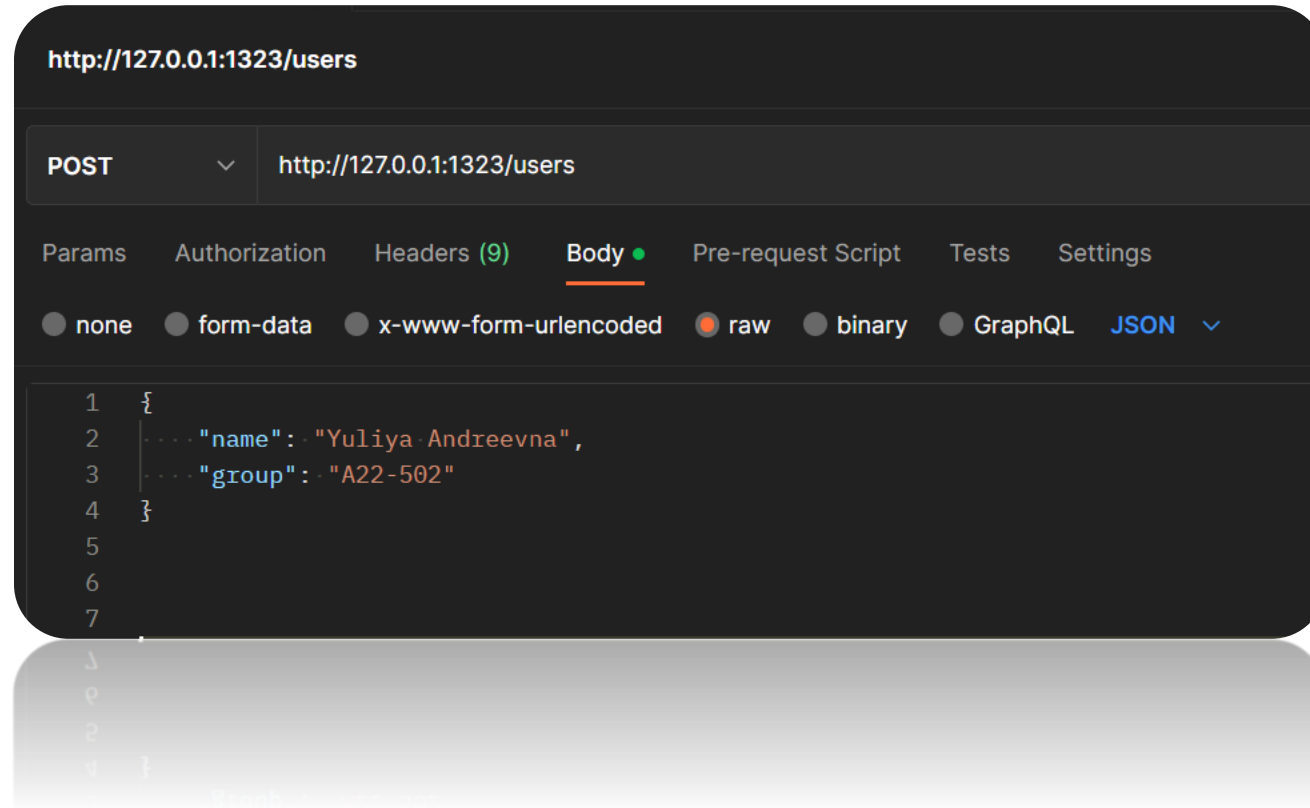


POSTMAN



# Тестирование

Создание юзера







# Тестирование

Просмотр юзеров

```
http://127.0.0.1:1323/users

GET http://127.0.0.1:1323/users

Params Authorization Headers (7) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON

Body Cookies Headers (3) Test Results
Pretty Raw Preview Visualize JSON

1 {
2   {
3     "id": 8,
4     "name": "Yuliya Andreevna",
5     "group": "A22-502"
6   },
7   {
8     "id": 9,
9     "name": "Marat Aidarovich",
10    "group": "M22-512"
11  },
12  {
13    "id": 10,
14    "name": "Random student",
15    "group": "B22-503"
16  },
17  {
18    "id": 11,
19    "name": "Random student 2",
20    "group": "B21-503"
21  }
22 }
```

```
33 {
34   {
35     "id": 12,
36     "name": "Random student 3",
37     "group": "B21-503"
38   }
39 }
```



# Тестирование

Просмотр билетов экзамена

```
http://127.0.0.1:1323/exam_cards

GET http://127.0.0.1:1323/exam_cards

Params Authorization Headers (7) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL

Body Cookies Headers (3) Test Results
Pretty Raw Preview Visualize JSON ↕

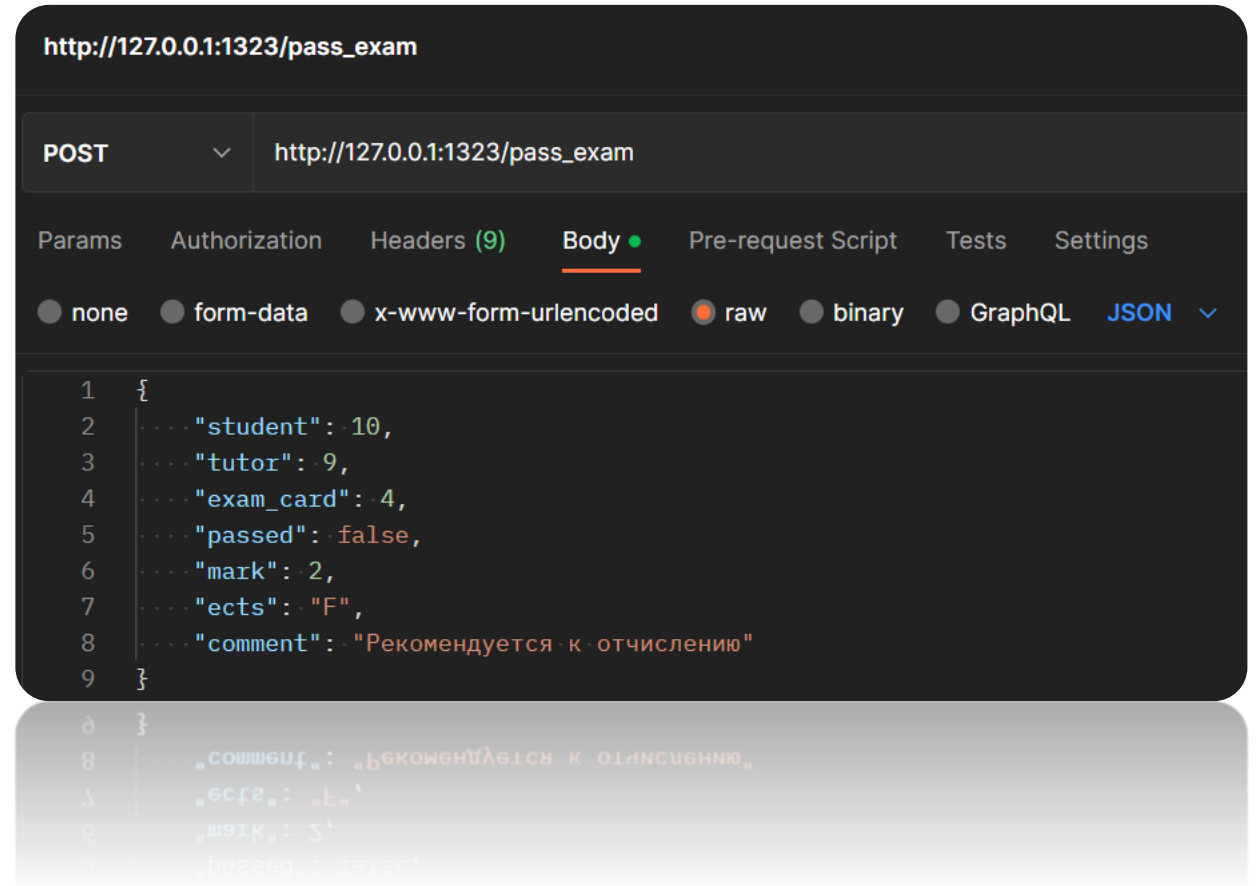
1 {
2   {
3     "id": 3,
4     "variant_number": 1,
5     "name": "Best card",
6     "task": "Напишите полное ФИО вашего преподавателя",
7     "date": "0001-01-01T03:00:00+03:00"
8   },
9   {
10    "id": 4,
11    "variant_number": 2,
12    "name": "Not so best card",
13    "task": "Напишите аналог приложения Wolfram Alpha",
14    "date": "0001-01-01T03:00:00+03:00"
15  },
16  {
17    "id": 5,
18    "variant_number": 3,
19    "name": "Not so best card again",
20    "task": "Напишите аналог приложения PhotoMath",
21    "date": "0001-01-01T03:00:00+03:00"
22  }
23 }
```

```
53 }
54
55 }
56
57 "date": "0001-01-01T03:00:00+03:00",
58 "task": "Напишите аналог приложения Wolfram Alpha",
59 "name": "Not so best card again",
60 "variant_number": 3
```



# Тестирование

Создание результата





# Тестирование

Просмотр результатов

GET http://127.0.0.1:1323/exams

Params Authorization Headers (7) Body Pre-request Script

Query Params

KEY
Key

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

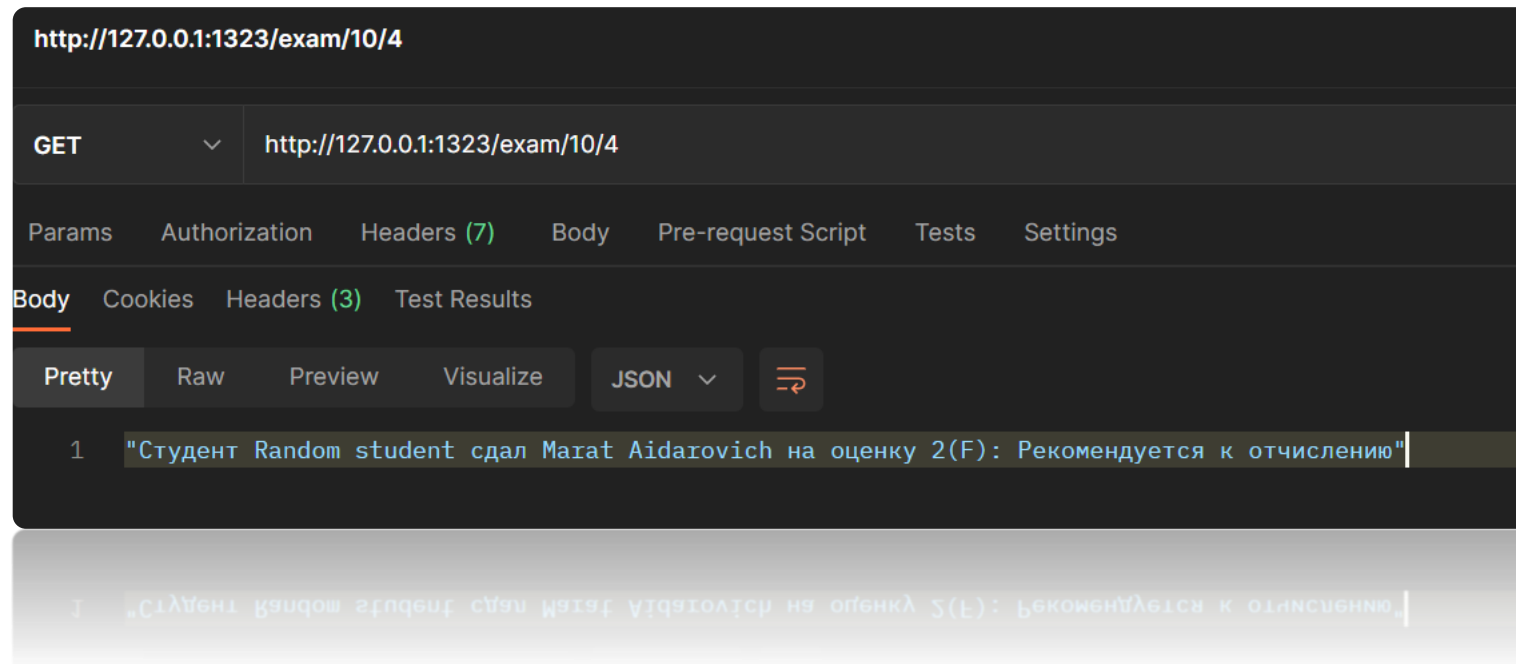
```
1 {
2   {
3     "student_id": 10,
4     "tutor_id": 9,
5     "exam_card_id": 4,
6     "passed": false,
7     "mark": 2,
8     "ects": "F",
9     "comment": "Рекомендуется к отчислению",
10    "user": {
11      "id": 0,
12      "name": "",
13      "group": ""
14    },
15    "tutor": {
16      "id": 0,
17      "name": "",
18      "group": ""
19    },
20    "examCard": {
21      "id": 0,
22      "variant_number": 0,
23      "name": "",
24      "task": "",
25      "date": "0001-01-01T00:00:00Z"
26    }
27  }
28 }
```

```
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
```



# Тестирование

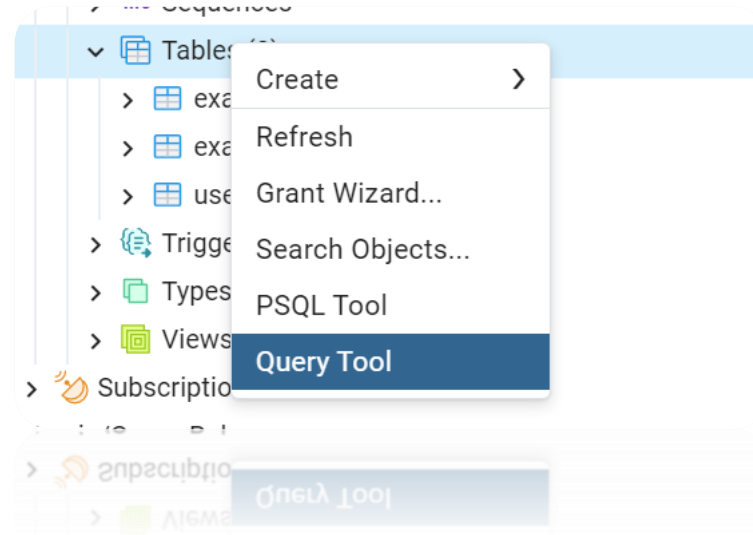
Просмотр форматированного результата





# Тестирование

pgAdmin





postgres/postgres@postgres

Query Scratch Pad x

```
1 select * from exams where student_id = 10 and exam_card_id = 4
2
3
```

Data Output Messages Notifications

	student_id [PK] bigint	tutor_id [PK] bigint	exam_card_id [PK] bigint	passed boolean	mark bigint	ects text	comment text
1	10	9	4	false	2	F	Рекомендуется к отчислению



~~А что это было сейчас~~

У меня очень много вопросов...







# А где найти проект?



[https://github.com/grigorevmp/Exam\\_process\\_web\\_application\\_MEPHI\\_NA](https://github.com/grigorevmp/Exam_process_web_application_MEPHI_NA)



That's all

