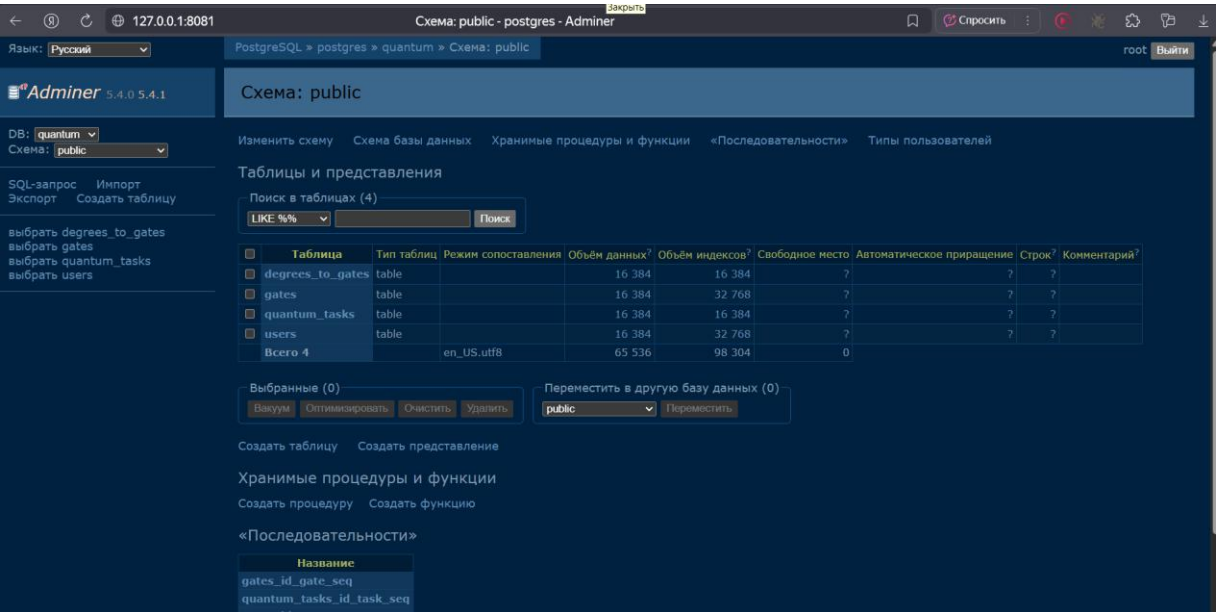


Формула для расчёта

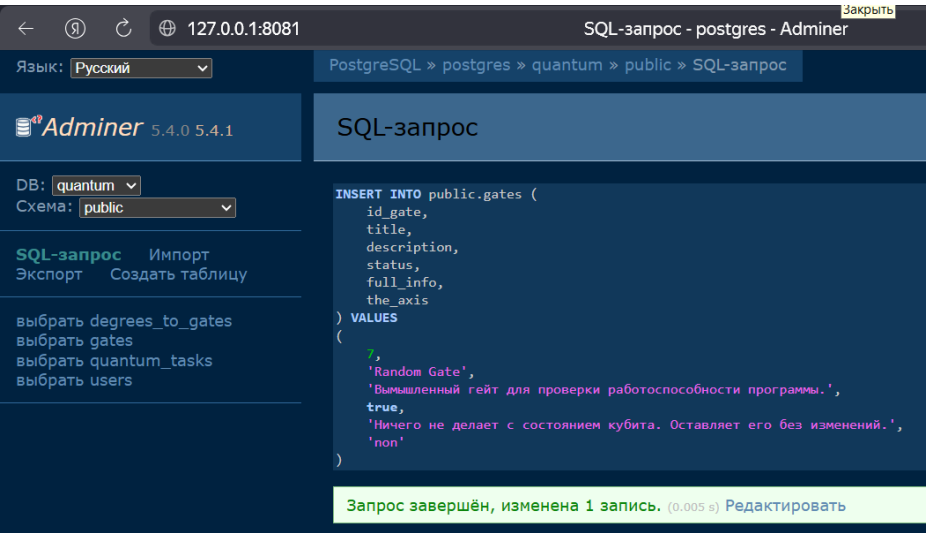
Общее значение для услуг в одной задаче – состояние кубита (исходное – нулевое)

Преобразования зависят от применённой карточки (формулы описаны в изображении гейта)

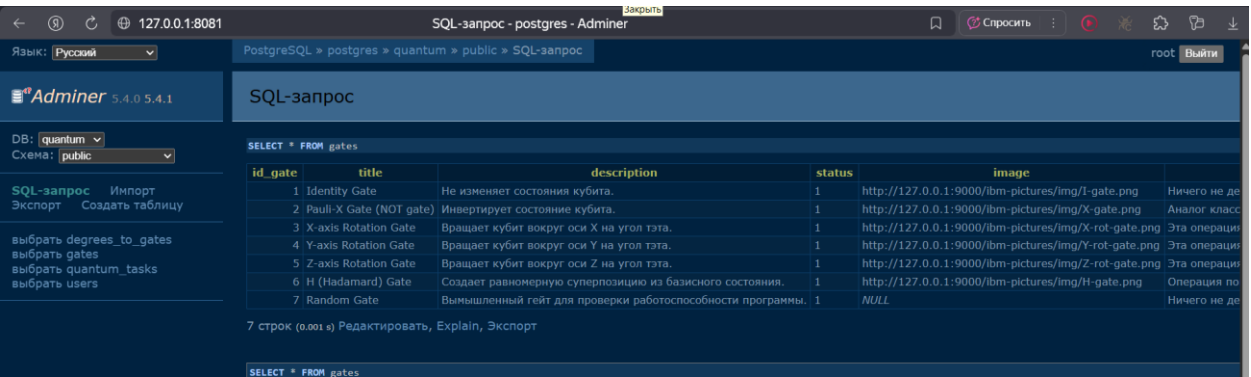
Adminer



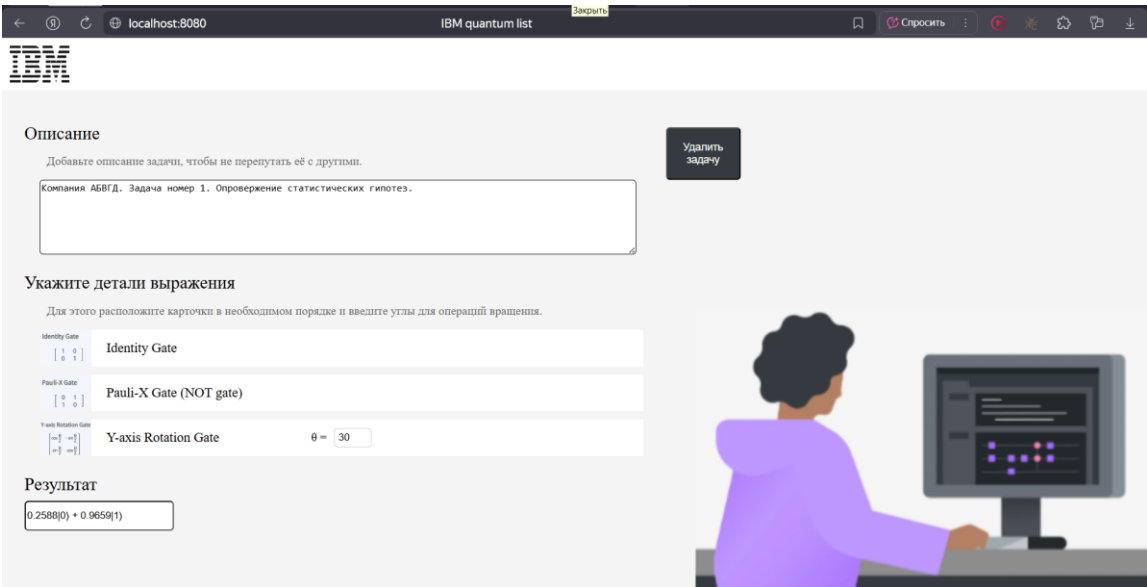
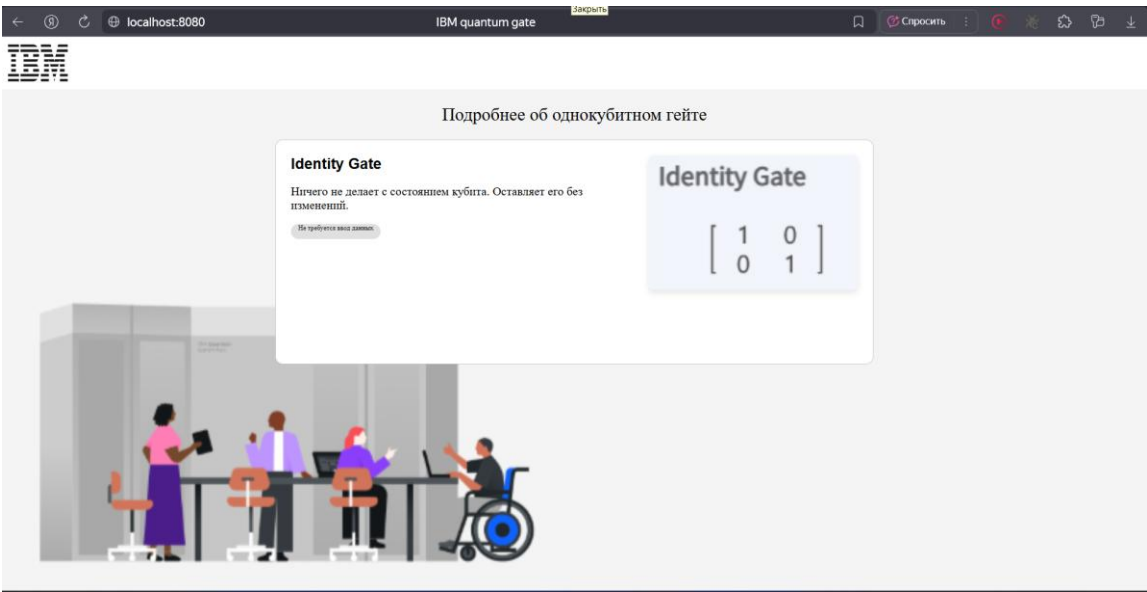
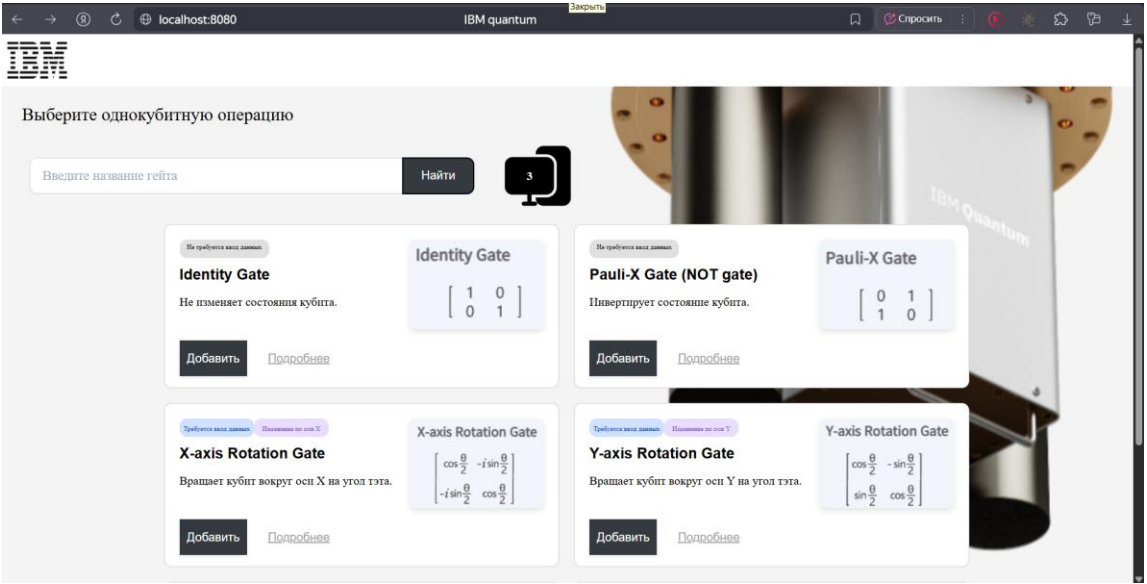
Добавление услуги



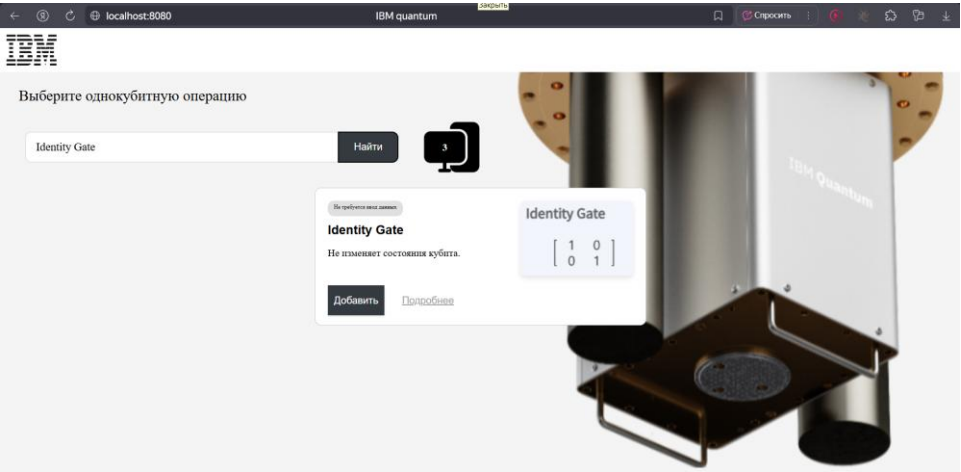
Просмотр результатов добавления



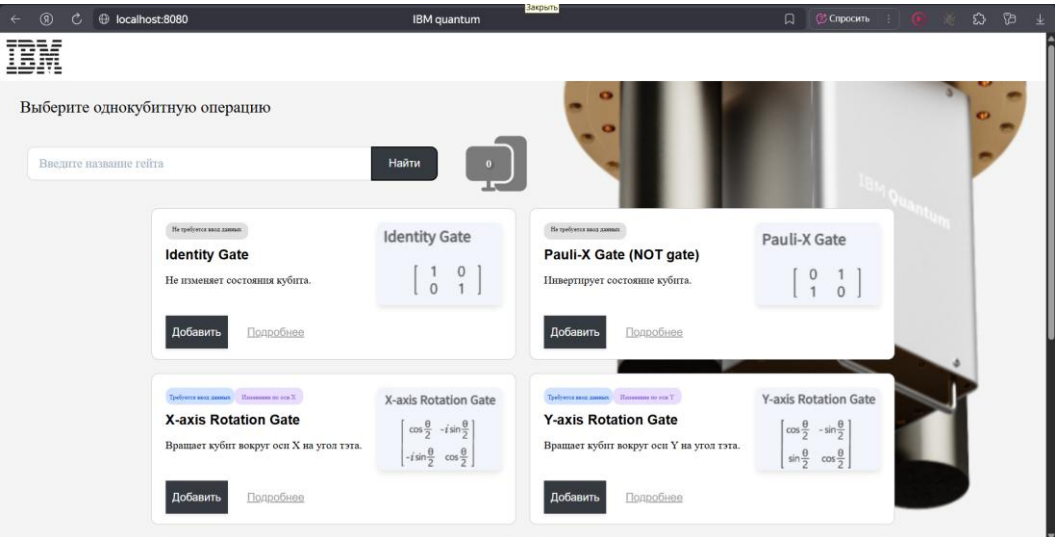
3-и страницы приложения



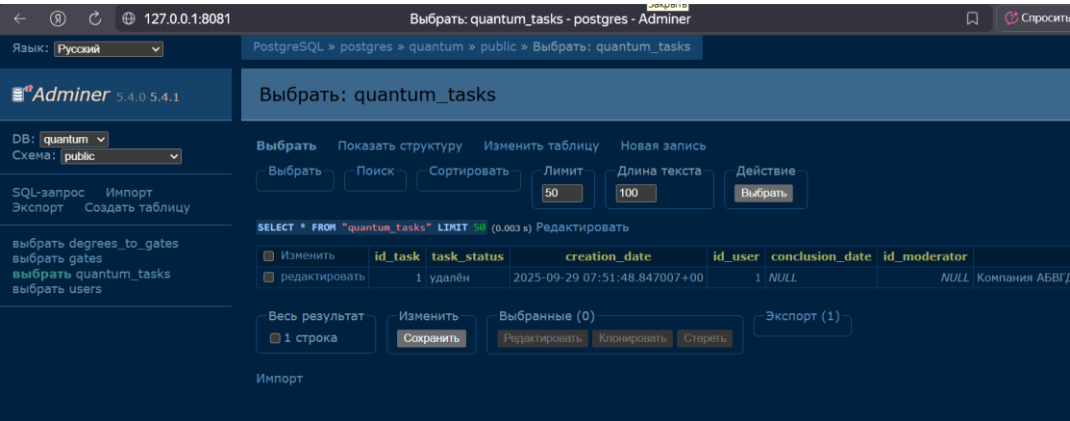
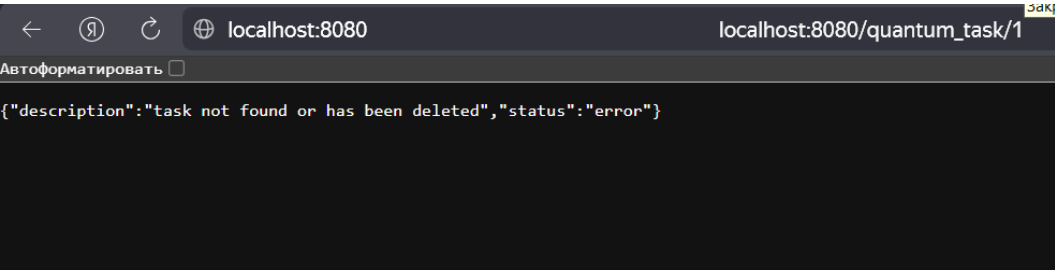
Поиск



Удаление заявки



Проверка удаления



Добавление услуг в заявку

←↻🌐localhost:8080IBM quantum list

IBM

Описание

Добавьте описание задачи, чтобы не перепутать её с другими.

Удалить задачу

Укажите детали выражения

Для этого расположите карточки в необходимом порядке и введите углы для операций вращения.

Pauli-X Gate

$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Pauli-X Gate (NOT gate)

Identity Gate

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Identity Gate

Hadamard Gate

$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

H (Hadamard) Gate

Z-axis Rotation Gate

$\begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$

Z-axis Rotation Gate

θ = 30


Y-axis Rotation Gate

$\begin{bmatrix} e^{-i\theta/2} & -i \\ i & e^{-i\theta/2} \end{bmatrix}$

Y-axis Rotation Gate

θ = 40

Результат



Проверка

←↻🌐127.0.0.1:8081Выбор: quantum_tasks - postgres - Adminer

Язык: Русский

PostgreSQL » postgres » quantum » public » Выбор: quantum_tasks

root Выйти

Adminer 5.4.0 5.4.1

DB: quantumСхема: public

SQL-запросИмпортЭкспортСоздать таблицу

выбрать degrees_to_gates
выбрать gates
выбрать quantum_tasks
выбрать users

Выбор: quantum_tasks

ВыбратьПоказать структуруИзменить таблицуНовая запись

Выбрать

Поиск

Сортировать

Лимит50

Длина текста100

ДействиеВыбрать

SELECT * FROM "quantum_tasks" LIMIT (0.001 s) Редактировать

<input type="checkbox"/>	Изменить	id_task	task_status	creation_date	id_user	conclusion_date	id_moderator	task_description
<input type="checkbox"/>	редактировать	1 удалён	2025-09-29 07:51:48.847007+00	1 NULL	0001-01-01 00:00:00+00	NULL	Компания АБВГД. Задача номер 1. Опровержение	
<input type="checkbox"/>	редактировать	2 удалён	0001-01-01 00:00:00+00	1 0001-01-01 00:00:00+00	0001-01-01 00:00:00+00	NULL		
<input type="checkbox"/>	редактировать	3 удалён	0001-01-01 00:00:00+00	1 0001-01-01 00:00:00+00	0001-01-01 00:00:00+00	NULL		
<input type="checkbox"/>	редактировать	4 удалён	0001-01-01 00:00:00+00	1 0001-01-01 00:00:00+00	0001-01-01 00:00:00+00	NULL		
<input type="checkbox"/>	редактировать	5 удалён	0001-01-01 00:00:00+00	1 0001-01-01 00:00:00+00	0001-01-01 00:00:00+00	NULL		
<input type="checkbox"/>	редактировать	6 черновик	0001-01-01 00:00:00+00	1 0001-01-01 00:00:00+00	0001-01-01 00:00:00+00	NULL		

Весь результат6 строк

ИзменитьСохранить

Выбранные (0)РедактироватьКлонироватьСтереть

Экспорт (6)

Импорт

Изменение м-м таблицы из БД

←↻🌐127.0.0.1:8081SQL-запрос - postgres - Adminer

Язык: Русский

PostgreSQL » postgres » quantum » public » SQL-запрос

SQL-запрос

Adminer 5.4.0 5.4.1

DB: quantumСхема: public

SQL-запросИмпортЭкспортСоздать таблицу

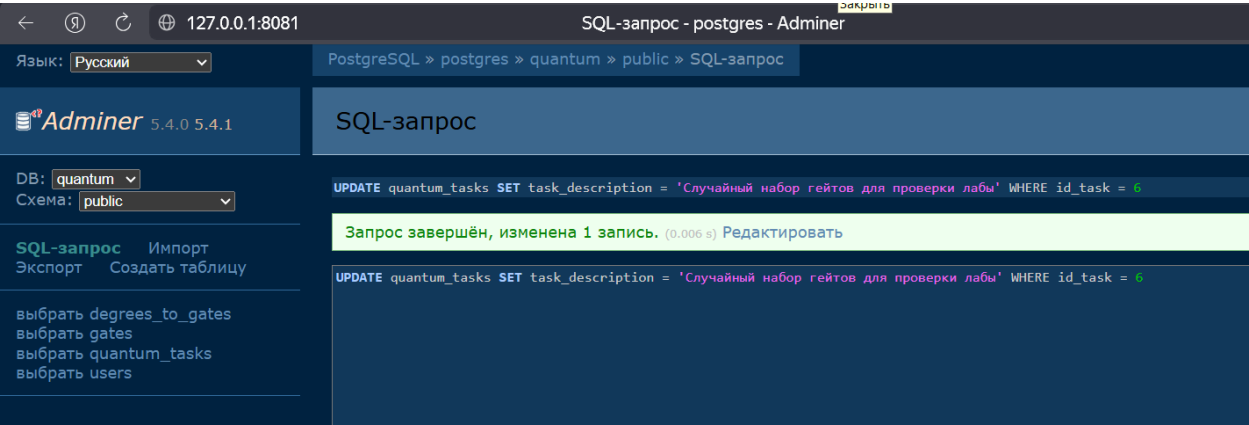
выбрать degrees_to_gates
выбрать gates
выбрать quantum_tasks
выбрать users

UPDATE degrees_to_gates SET degrees = 10 WHERE id_gate = 5 AND id_task = 6

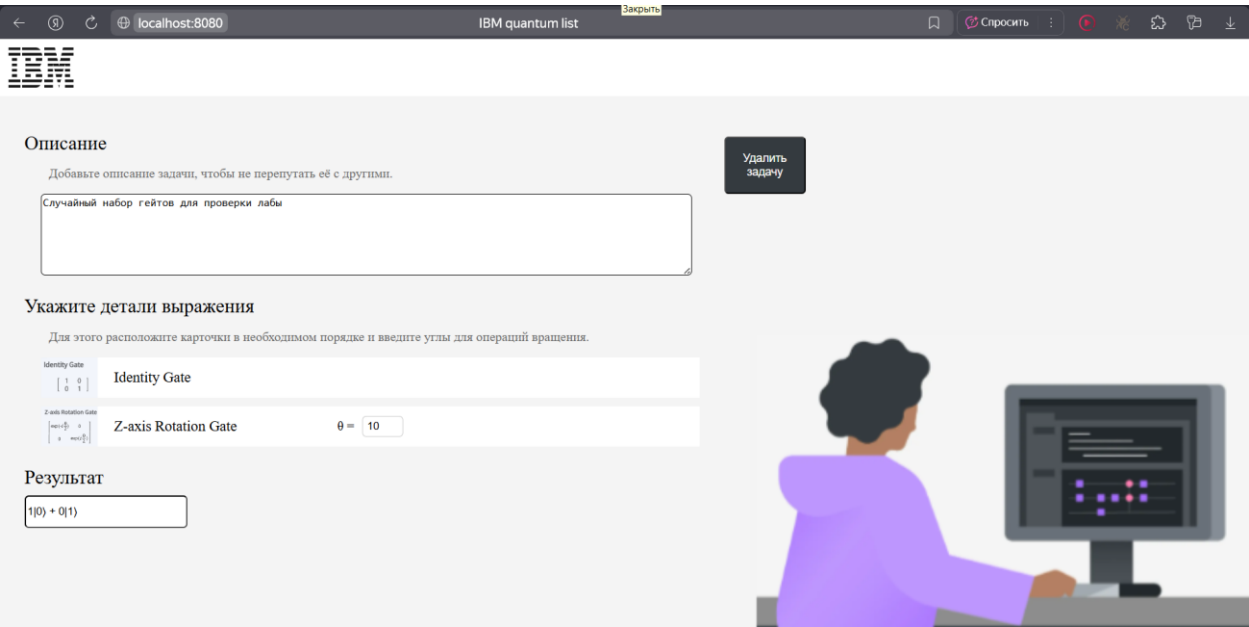
Запрос завершён, изменена 1 запись. (0.006 s) Редактировать

UPDATE degrees_to_gates SET degrees = 10 WHERE id_gate = 5 AND id_task = 6

Изменение таблицы заявки из БД



Результат изменений



Демонстрация кода

Модели

```
front_start > internal > app > ds > DegreesToGates.go > ...
1 package ds
2
3 type DegreesToGates struct {
4     ID_gate uint `gorm:"primaryKey;column:id_gate;not null"`
5     ID_task uint `gorm:"primaryKey;column:id_task;not null"`
6     Degrees *float32
7     // связи, необходимые для работы сервиса
8     Gate Gate `gorm:"foreignKey:ID_gate;references:ID_gate"`
9     Task QuantumTask `gorm:"foreignKey:ID_task;references:ID_task"`
10 }
11
```

т. m-m

```
front_start > internal > app > ds > Gate.go > Gate > Degrees
1 package ds
2
3 type Gate struct {
4     ID_gate uint `gorm:"column:id_gate;primaryKey"`
5     Title string `gorm:"column:title;size:255;not null;default:gate-no-name;unique"`
6     Description string `gorm:"column:description;not null"`
7     Status bool `gorm:"column:status;default:true"`
8     Image *string `gorm:"column:image"`
9
10 // subject area
11 FullInfo string `gorm:"column:full_info;not null"`
12 TheAxis string `gorm:"column:the_axis"`
13 /*
14     Удалил согласно правкам к ЛР 1
15     IsEditable bool `gorm:"default:false"`
16 */
17
18 // Несущая связь
19 Degrees []DegreesToGates `gorm:"foreignKey:ID_gate;references:ID_gate"`
20 }
```

т. услуги

```

front_start > internal > app > ds > QuantumTask.go > QuantumTask
1 package ds
2
3 import "time"
4
5 type QuantumTask struct {
6     ID_task uint `gorm:"column:id_task;primaryKey;autoincrement"`
7     TaskStatus string `gorm:"column:task_status;size:255;not null;default:черновики"`
8     CreationDate time.Time `gorm:"column:creation_date;not null"`
9     ID_user uint `gorm:"column:id_user;not null"`
10    ConclusionDate time.Time `gorm:"column:conclusion_date"`
11    ID_moderator *uint `gorm:"column:id_moderator"`
12
13    TaskDescription string `gorm:"column:task_description"`
14    Res_koeff_0 float32 `gorm:"column:res_koeff_0;default:11"`
15    Res_koeff_1 float32 `gorm:"column:res_koeff_1;default:0"`
16
17    // Несушая связь
18    GatesDegrees []DegreesToGates `gorm:"foreignKey:ID_task;references:ID_task"`
19    Moderator Users `gorm:"foreignKey:ID_moderator;references:ID_user"`
20    // Связь для программы
21    User Users `gorm:"foreignKey:ID_user;references:ID_user"`
22 }

```

Т. заказы

```

front_start > internal > app > ds > Users.go > ...
1 package ds
2
3 type Users struct {
4     ID_user uint `gorm:"column:id_user;primaryKey"`
5     Login string `gorm:"column:login;not null;size:255;unique"`
6     Password string `gorm:"column:password;size:255;not null"`
7     IsAdmin bool `gorm:"column:is_admin;default:false"`
8     // Связь несущая
9     Tasks []QuantumTask `gorm:"foreignKey:ID_user;references:ID_user" json:"-"`
10 }

```

Т. пользователи

Составной уникальный ключ

```

front_start > internal > app > ds > DegreesToGates.go > ...
1 package ds
2
3 type DegreesToGates struct {
4     ID_gate uint `gorm:"primaryKey;column:id_gate;not null"`
5     ID_task uint `gorm:"primaryKey;column:id_task;not null"`
6     Degrees *float32
7     // Связи, необходимые для работы сервиса
8     Gate Gate `gorm:"foreignKey:ID_gate;references:ID_gate"`
9     Task QuantumTask `gorm:"foreignKey:ID_task;references:ID_task"`
10 }
11

```

4-ре контроллера через ORM

```

func (r *Repository) GetGates() ([]ds.Gate, error) {
    var gates []ds.Gate

    err := r.db.Find(&gates).Error
    if err != nil {
        return nil, err
    }

    if len(gates) == 0 {
        return nil, fmt.Errorf("gates not found")
    }
    return gates, nil
}

func (r *Repository) GetGatesByName(title string) ([]ds.Gate, error) {
    var gates []ds.Gate
    err := r.db.Where("title ILIKE ?", "%"+title+"%").Find(&gates).Error
    if err != nil {
        return nil, err
    }
    return gates, nil
}

// GetTaskWithGates получает задачу со всеми связанными гейтами и их углами поворота.
func (r *Repository) GetTaskWithGates(taskID uint) (*ds.QuantumTask, error) {
    var task ds.QuantumTask

    // Используем Preload для загрузки связанных данных через связующую таблицу
    err := r.db.
        Preload("GatesDegrees.Gate").
        First(&task, taskID).
        Error

    if err != nil {
        if errors.Is(err, gorm.ErrRecordNotFound) {
            return nil, errors.New("task not found")
        }
        return nil, fmt.Errorf("error fetching task: %v", err)
    }

    // Проверяем, что задача не удалена (адаптируйте под ваши статусы)
    if task.TaskStatus == ds.StatusDeleted {
        return nil, errors.New("task not found or has been deleted")
    }

    return &task, nil
}

```

```

// AddGateToTask добавляет гейт в задачу (создает запись в таблице m-m).
func (r *Repository) AddGateToTask(taskID, gateID uint) error {
    // Проверяем, нет ли уже такого фактора в заявке, чтобы избежать дублей
    var count int64
    r.db.Model(&ds.DegreesToGates{}).Where("id_task = ? AND id_gate = ?", taskID, gateID).Count(&count)
    if count > 0 {
        return errors.New("gate already in task")
    }

    link := ds.DegreesToGates{
        ID_task: taskID,
        ID_gate: gateID,
    }
    return r.db.Create(&link).Error
}

```

```

func (r *Repository) GetGateByID(id int) (*ds.Gate, error) {
    var gate ds.Gate
    err := r.db.First(&gate, id).Error
    if err != nil {
        return nil, err
    }
    return &gate, nil
}

```

Удаление заявки через SQL

```
// LogicallyDeleteTask выполняет логическое удаление заявки через чистый SQL UPDATE.
func (r *Repository) LogicallyDeleteTask(taskID uint) error {
    // Используем Exec для выполнения "сырого" SQL-запроса
    result := r.db.Exec("UPDATE quantum_tasks SET task_status = ? WHERE id_task = ?", ds.StatusDeleted, taskID)
    return result.Error
}
```