Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Поволжский государственный университет телекоммуникаций и информатики

Факультет №1

Кафедра «Программная инженерия» (ПрИ)

ДОПУСТИТЬ К ЗАЩИТЕ ЗАЩИЩЕН (А) С ОЦЕНКОЙ\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ Руководитель \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО) (подпись) (ФИО)

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_

(дата) (дата)

КУРСОВАЯ РАБОТА

по дисциплине Прикладное программирование

наименование дисциплины (модуля)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

наименование темы (при наличии)

ВЫПОЛНИЛ (А)

студент (ка) ПрИ-21 Старостин А.А.

(группа) (ФИО)

220357

(№ зачетной книжки)

Самара 2024

# Рецензия

на курсовую работу по дисциплине Прикладное программирование

Студента Старостина А.А.

Рецензент – руководитель КР:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О., степень, звание, должность)

*В рецензии отражается соответствие КР утвержденному индивидуальному заданию, глубина проработки поставленных вопросов. Выявляются положительные стороны проделанной работы, а также ее недостатки*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Результат рецензирования: допущен/не допущен к защите

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО руководителя)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

# Задание

на курсовую работу по дисциплине Прикладное программирование

Студента Старостина А.А.

Рецензент – руководитель КР:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О., степень, звание, должность)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО руководителя)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

# Содержание

[Рецензия 2](#_Toc181960047)

[Задание 3](#_Toc181960048)

[Содержание 4](#_Toc181960049)

[Введение 5](#_Toc181960050)

[Описание методов и технологий 6](#_Toc181960051)

[Цели и задачи работы 7](#_Toc181960052)

[Реализация 8](#_Toc181960053)

[Тестирование 9](#_Toc181960054)

[Заключение 10](#_Toc181960055)

[Список использованных источников 11](#_Toc181960056)

# Введение

В данном проекте будет описан процесс разработки, его этапы и средства разработки, но для начала нужно определить, почему трекинг времени является довольно интересной задачей.

Часто при задумке проектов в любой сфере, его создание делится на выполнение иерархии задач, которые его составляют, и все подобные задачи имеют один главный ресурс- время. Важно понимать, за какие задачи нужно взяться раньше и частью какой более крупной задачи они являются. Также важна оценка продуктивности за период для анализа своих возможностей. Все подобные функции и представляют системы трекинга времени.

# Цели и задачи работы

**Цель работы:** создать клиент-серверное приложение трекинга времени с использованием средств языка Golang и баз данных.

Задачи работы:

* Создать БД с необходимыми таблицами;
* Создать ограничения БД (ограничение целостности, связи таблиц, ограничения значений атрибутов);
* Создать триггеры БД для заполнения поддерживающий таблиц и таблиц для отчётности;
* Создать сервер с необходимыми маршрутам;
* Создать авторизацию на сервере для распределения задач между пользователями;
* Создать способ представление задач, иерархии, отчётов;
* Создать веб-страницу для работы с клиентом;
* Создать ограничения по вводу данных пользователем;
* Создать обращения к маршрутам сервера.

# Описание методов и технологий

Использованные технологии БД:

* Postgresql 17
* pgAdmin4

Технологии для создания сервера:

* Язык программирования Golang;
* Библиотека "github.com/golang-jwt/jwt/v4";
* Библиотека "database/sql";
* Библиотека "encoding/base64";
* Библиотека "encoding/json".

Технологии для создания клиентской части:

* Язык разметки html;
* Язык таблиц стилей CSS;
* Язык программирования JavaScript;
* Библиотека JS для работы с графическими элементами "https://d3js.org/d3.v4.min.js».

# Реализация

Первой задачей, которая должны быть выполнена, это создание БД.

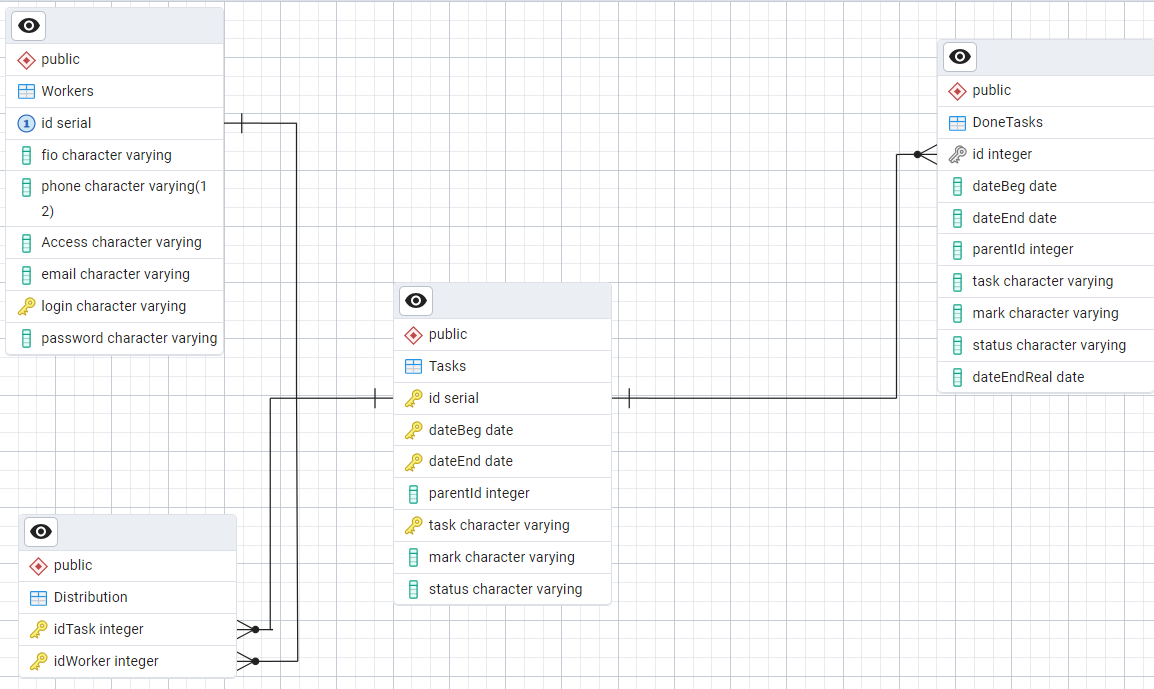
**ERD-Диаграмма**

Рисунок 1 – «ERD-Диаграмма БД»

Начать пояснение стоит с того, что связь владельца с задачей в теории будет иметь связь «М:М», поэтому была создана поддерживающая таблица «Distribution» для распределения.

Таблица «Tasks» имеет текст, даты, отметки, а также рекурсивный атрибут связи с родительской записью «parentId» для создания иерархии.

Таблица «DoneTasks» используется в целях создания отчётов, данные в неё добавляются автоматически при завершении задачи пользователем.

Для значений индексов стандартное значение высчитывают функции, дающая ближайшее новое значение, примерный вид:

nextval('"Workers\_id\_seq"'::regclass)

**Триггеры**

«done\_del» - запускается при изменениях в таблице «Tasks» и если статус готовой задачи убрали, то удаляет запись изменённой задачи.

BEGIN

IF NEW."status"<>'done' THEN

DELETE from public."DoneTasks"

where id=NEW.id;

END IF;

RETURN NEW;

END;

«done\_func» - запускается при создании или изменении в таблице «Tasks» и если задача со статусом готовой, она добавляется в таблицу готовых задач.

BEGIN

IF (NEW."status"='done') THEN

INSERT INTO

public."DoneTasks"(id,"dateBeg","dateEnd","parentId","task","mark","status")

VALUES

(NEW.id,NEW."dateBeg",NEW."dateEnd",NEW."parentId",NEW."task",NEW."mark",NEW."status");

END IF;

RETURN NEW;

END;

«new\_usr\_task0» - запускается при создании нового пользователя, цель триггера добавить новому пользователю корневую задачу, т. к. начало нужно будет отслеживать, а значение «NULL» в json формат зашифровать нельзя.

BEGIN

INSERT INTO public."Distribution"(

"idTask", "idWorker")

VALUES (0, NEW.id);

RETURN NEW;

END;

**Запросы**

Следующие запросы будут использоваться в маршрутах сервера.

Запрос на получение задач с определённым остатком времени

WITH LeftD as (SELECT id, task, mark, status, "dateEnd"-"dateBeg" as daysHad, "dateEnd"-CURRENT\_DATE as daysLeft

FROM public."Tasks"),usr as(select id from public."Workers" where "login"=$1),

his as(select "idTask" from public."Distribution" where "idWorker"= (select \* from usr))

select \* from LeftD where daysLeft<=$2 and status not like 'done' and id in (select \* from his) order by daysLeft asc

Запрос на получение задач, которые с меткой выполнено, за последний промежуток времени

WITH LeftD as (SELECT id, task, mark, status, CURRENT\_DATE-"dateEnd" as daysPassed

FROM public."Tasks"),usr as(select id from public."Workers" where "login"=$1),

his as(select "idTask" from public."Distribution" where "idWorker"= (select \* from usr))

select distinct \* from LeftD where daysPassed between 0 and $2 and status like 'done' and id in (select \* from his) order by daysPassed asc

Запрос на открытие доступа к задаче новому человеку

WITH usr as(select id from public."Workers" where "login"=$1)

INSERT INTO public."Distribution"(

"idTask", "idWorker")

VALUES ($2, (select \* from usr));

Запрос на удаление задачи

DELETE FROM public."Tasks"

WHERE id=$1;

Запрос на изменение задачи

UPDATE public."Tasks"

SET "dateBeg"=$1, "dateEnd"=$2, "parentId"=$3, task=$4, mark=$5, status=$6

WHERE id=$7;

Серия запросов на создание задачи

INSERT INTO public."Tasks"(

"dateBeg", "dateEnd", "parentId", task, mark, status)

VALUES ( $1, $2, $3, $4, $5, $6)

RETURNING "id";

WITH usr as(select id from public."Workers" where "login"=$2)

INSERT INTO public."Distribution"(

"idTask", "idWorker")

VALUES ($1, (select \* from usr limit 1));

Запрос на поиск логина и пароля для авторизации

SELECT p.login,p.password

FROM public."Workers" as p

where (p.email=$1 and p.password=$2) or (p.login=$1 and p.password=$2);

Серия запросов для регистрации

SELECT id

FROM public."Workers"

WHERE login=$1;

INSERT INTO public."Workers"(

fio, email, login, password)

VALUES ($1, $2, $3, $4);

Запрос с рекурсией для поиска дочерних записей

with recursive whosYourChildren("parentId", id,depth) as (

--start with the "anchor" row

select

"parentId", id,0

from public."Tasks" as t

where

id = $1 --parameterize me

union all

select

t."parentId", t.id, whosYourChildren.depth + 1

from public."Tasks" as t

join whosYourChildren on whosYourChildren.id = t."parentId"

),

allt("parentId", id,depth, "status") AS (

select whosYourChildren."parentId",whosYourChildren.id,whosYourChildren.depth,t.status

from whosYourChildren, public."Tasks" as t

where whosYourChildren."parentId" is not null and depth>0 and t.id=whosYourChildren.id

order by

whosYourChildren.depth

)

,BYAUTH AS ( select distinct t.id

from public."Tasks" as t, public."Distribution" as d,public."Workers" as w

where (t.id= d."idTask" and d."idWorker"=w.id and w.login=$1) or t.id=0)

select distinct \* from allt where allt.id in (select \* from BYAUTH)

Запрос с рекурсией на поиск родителей задачи

with recursive whosYourDaddy("parentId", id,depth) as (

--start with the "anchor" row

select

"parentId", id,1

from public."Tasks" as t

where

id = $1 --parameterize me

union all

select

t."parentId", t.id, whosYourDaddy.depth + 1

from public."Tasks" as t

join whosYourDaddy on whosYourDaddy."parentId" = t.id

)

select

\*

from whosYourDaddy

where "parentId" is not null

order by

whosYourDaddy.depth;

Серия запросов для поиска задач по родителю

SELECT \* from public."Tasks" where "parentId" IS NOT NULL AND "parentId"=$1

SELECT distinct w."login"

FROM public."Distribution" as d, public."Workers" as w

WHERE d."idWorker" = w.id and d."idWorker"=1 and d."idTask"=$1

**Сервер**

Начальная настройка и запуск, создание глобальных переменных

var connStr string = "user=postgres password=1 dbname=pp sslmode=disable"

var logFile, \_ = os.OpenFile("app.log", os.O\_APPEND|os.O\_WRONLY|os.O\_CREATE, 0600)

func main() {

listenAddr := os.Getenv("LISTEN\_ADDR")

if len(listenAddr) == 0 {

listenAddr = ":8080"

}

mux := http.NewServeMux()

setupHandlers(mux)

log.Fatal(http.ListenAndServe(listenAddr, mux))

log.SetOutput(logFile)}

**Объявление маршрутов для сервера**

func setupHandlers(mux \*http.ServeMux) {

mux.HandleFunc("/", middleware)

mux.HandleFunc("/login", login)

mux.HandleFunc("/registration", registration)

mux.HandleFunc("/selectAll", selectAll)

mux.HandleFunc("/data", data1)

mux.HandleFunc("/selectByParent", selectByParent)

mux.HandleFunc("/children", children)

mux.HandleFunc("/parents", parents)

mux.HandleFunc("/createTask", createTask)

mux.HandleFunc("/changeTask", changeTask)

mux.HandleFunc("/deleteTask", deleteTask)

mux.HandleFunc("/task/findByTask", findByTask)

mux.HandleFunc("/task/addWorker", addWorker)

mux.HandleFunc("/report/left", RepLeft)

mux.HandleFunc("/report/forLast", RepforLast)

fmt.Println("Сервер запущен")

}

Настройка начальных логов для всех маршрутов, как и у всех остальных маршрутов в параметрах функции http.ResponseWriter и \*http.Request для получения тела и заголовков запроса и ссылка для отправки ответов

func middleware(w http.ResponseWriter, req \*http.Request) {

time := time.Now().Format(time.ANSIC)

log.Printf("\nExecuting middleware at: %v Method:%s URL:%s\n", time, req.Method, req.URL.Path)

log.SetOutput(logFile)

}

**Построение маршрутов**

Обработка http запроса начинается с разрешения на использование файлов сервера и клиента с одного компьютера, т. к. при разработке и отладке на одном и том же компьютере вызывает ошибку от JavaScript

w.Header().Set("Access-Control-Allow-Origin",

w.Header().Set("Access-Control-Allow-Methods", "GET, POST, OPTIONS")

w.Header().Set("Access-Control-Allow-Headers", "Content-Type, Token, Authorization")

if req.Method == http.MethodOptions {

return

}

Далее в запросах нужно получить токен сессии пользователя, проверить его на валидность и узнать логин для последующих запросов к БД

// Получаем токен из заголовков запроса

tokenString := req.Header.Get("Authorization")

// Проверяем токен

cl, err := validateToken(tokenString) //cl

if err != nil {

http.Error(w, " BAD TOKEN", http.StatusUnauthorized)

return

}

//Узнаём кто сделал запрос

claims, \_ := cl.(jwt.MapClaims)

sub := claims["sub"]

Далее идёт проверка метода и подключение к БД, в случае правильного метода

if req.Method == "POST" { db, err := sql.Open("postgres", connStr)

if err != nil {

http.Error(w, " Database died<:(|", http.StatusServiceUnavailable)

log.Println(err)

panic(err)

}

defer db.Close()…}

Если запрос подразумевает тело с параметрами, дальше идёт парсинг тела запроса в структуру

var data ParentIdForm

err = json.NewDecoder(req.Body).Decode(&data)

if err != nil {

http.Error(w, "Ошибка парсинга JSON данных: "+err.Error(), http.StatusBadRequest)

return

}

После идёт запрос к БД, где «$» обозначает параметр, и после строки текста запроса нужно через запятую указывать параметры

result, err := db.Query(`…..$1`,par1)

if err != nil {

fmt.Println("Ошибка выполнения запроса:", err)

return

}

Для обхода результата запроса используется метод массивов .Next(), значения загружаются в переменную через .Scan({переменная})

for result.Next() {

t := LeftStruct{}

err := result.Scan(&t.Id, &t.Task, &t.Mark, &t.Status, &t.DaysHad, &t.DaysLeft)}

Для создания ответа клиенту мы кодируем данные через json.Marshal(), далее мы устанавливаем заголовок запросу и через .Write() отправляем клиенту

jsonResponse, err := json.Marshal(datamass)

if err != nil {

fmt.Println("Ошибка создания ответа:", err)

return

}

w.Header().Set("Content-Type", "application/json")

w.Write(jsonResponse)

**Клиент**

Теперь рассмотрим клиентские функции, и начать стоит с сохранения и загрузки из хранилища сессии, и делается это с помощью sessionStorage.setItem() и sessionStorage.getItem()

const token = JSON.parse(sessionStorage.getItem("JWT"));

sessionStorage.setItem("JWT", JSON.stringify(data));

Для распределения функций между кнопками использован jQueuery

$('#{ID}).click({FUNC NAME});

Запросы к БД происходят с помощью функции fetch, и функция не заканчивается пока не получит ответ или ошибку с помощью catch функций

var url = "http://localhost:8080/report/forLast";

let days= document.getElementById("RepDays").value;

var data = { pi: Number(days) };

fetch(url, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'Authorization': token.tokenStr,

},

body: JSON.stringify(data),

})

.then(response => response.json())

.then(data => {DoneFor2(data);})

.catch(error => console.error(error));

После получения ответа из запроса мы получаем структуру с полями, что мы указывали на сервере и можем без какой либо обработки их использовать.

При запросе на задачи они выводятся под определённым шаблонов на страницу как отдельные объекты, и запрос на создания таких представлений из данный сервера выглядит следующим образом.

document.querySelectorAll('.taskInstance').forEach(e => e.remove()); //очистка старых значений

data.forEach(function(entry) {

let div = document.createElement('div');

div.className = "taskInstance";

let table = document.createElement('table');

table.className = "taskInside";

let headerDate = document.createElement('tr');

let Dbeg = document.createElement('th');

let Dend = document.createElement('th');

Dbeg.innerHTML=('Дата нач.');

Dend.innerHTML=('Дата кон.');

headerDate.appendChild(Dbeg);

headerDate.appendChild(Dend);

table.appendChild(headerDate);

let dataDate = document.createElement('tr');

Dbeg = document.createElement('td');

Dend = document.createElement('td');

Dbeg.innerHTML=entry.TaskF.dateBeg.slice(0,10);

Dend.innerHTML=entry.TaskF.dateEnd.slice(0,10);

dataDate.appendChild(Dbeg);

dataDate.appendChild(Dend);

table.appendChild(dataDate);

let headerResp = document.createElement('tr');

let resp = document.createElement('th');

resp.setAttribute('colspan','2');

resp.innerHTML=('Ответственные');

headerResp.appendChild(resp);

table.appendChild(headerResp);

let dataResp = document.createElement('tr');

if (entry.WorkersF.length!=0) {

resp = document.createElement('td');

resp.setAttribute('colspan','2');

entry.WorkersF.forEach(function(worker){

resp.innerHTML+=worker.login+ "<br>";

});

} else {

resp = document.createElement('td');

resp.setAttribute('colspan','2');

resp.innerHTML='Не назначенно';

}

dataResp.appendChild(resp);

table.appendChild(dataResp);

let headerTask = document.createElement('tr');

let task = document.createElement('th');

task.setAttribute('colspan','2');

task.innerHTML=('Задача');

headerTask.appendChild(task);

table.appendChild(headerTask);

let dataTask = document.createElement('tr');

task = document.createElement('td');

task.setAttribute('colspan','2');

task.innerHTML=entry.TaskF.task;

dataTask.appendChild(task);

table.appendChild(dataTask);

let headerMarks = document.createElement('tr');

let marks = document.createElement('th');

marks.setAttribute('colspan','2');

marks.innerHTML=('Метки');

headerMarks.appendChild(marks);

table.appendChild(headerMarks);

let dataMarks = document.createElement('tr');

marks = document.createElement('td');

marks.innerHTML=entry.TaskF.mark;

dataMarks.appendChild(marks);

marks = document.createElement('td');

marks.innerHTML=entry.TaskF.status;

dataMarks.appendChild(marks);

table.appendChild(dataMarks);

let taskid= document.createElement('p');

taskid.innerHTML=entry.TaskF.id;

taskid.className = "taskNumP";

let button = document.createElement('button');

button.className = "taskInB";

button.textContent = 'Перейти ввнутрь';

let button2 = document.createElement('button');

button2.className = "taskChange";

button2.textContent = 'Изменить';

let button3 = document.createElement('button');

button3.className = "taskDelete";

button3.textContent = 'Удалить';

let button4 = document.createElement('button');

button4.className = "taskAddWorker";

button4.textContent = 'Добавить человека в задачу';

div.appendChild(taskid);

div.appendChild(button);

div.appendChild(button2);

div.appendChild(button3);

div.appendChild(button4);

div.appendChild(table);

div.id=entry.TaskF.id;

$('.tasksContainer').append(div);

});

$('.taskInB').click(selectcInside);

$('.taskChange').click(ChangeTask);

$('.taskDelete').click(DeleteTask);

$('.taskAddWorker').click(taskAddWorker);

После выполнение запроса к БД страница заполниться объектами следующего вида (рис. 2)

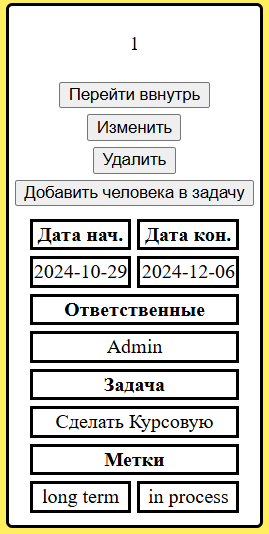


Рисунок 2 – «Шаблонный объект на веб-странице»

Далее было создано графическое навигационное меню (рис. 3) с помощью пакета JS .d3, оно использует запрос о родительских и дочерних записях, который был разобран ранее, для него необходим список номеров задач «nodes» и «link» в виде «1->2», также при событии неведения на «node» идёт получение её номера и вывод информации, при нажатии же мы переходим внутрь этой задачи

var data ={nodes: nodes,links: links};

var svg = d3.select("svg"),

width = +svg.attr("width"),

height = +svg.attr("height");

var simulation = d3.forceSimulation()

.force("link", d3.forceLink().id(function(d) { return d.id; }))

.force("charge", d3.forceManyBody())

.force("center", d3.forceCenter(width / 2, height / 2));

var link = svg.selectAll(".link")

.data(data.links)

.enter().append("line")

.attr("class", "link");

var node = svg.selectAll(".node")

.data(data.nodes)

.enter().append("circle")

.attr("class", "node")

.attr("r", 5)

.attr("fill", function(d) {

if (d.id == indTek ) {

return 'grey'; // Цвет для неактивного статуса

}else if (d.id === 0) {

return 'fuchsia'; // Цвет для активного статуса

}

else if (d.status === 'done') {

return 'green'; // Цвет для активного статуса

} else if (d.status === 'in process') {

return 'yellow'; // Цвет для неактивного статуса

}else if ((d.status === 'inactive' ) || (d.status === 'waiting' )) {

return 'red'; // Цвет для неактивного статуса

}

else {

return 'black'; // Цвет по умолчанию

}

})

.on("mouseover", function(d) {

let dialogTask=document.getElementById("dialogTask")

dialogTask.innerHTML="Задача #" + d.id

dialogTask.style.top=`${d.y+30}px`

dialogTask.style.left=`${d.x+15}px`

show(dialogTask);

})

.on("mouseout", function(d) {

hide(dialogTask);

})

.on("click", function(d) {

hide(dialogTask);

selectcInside(d.id);

s.innerHTML = ` <defs>

<linearGradient id="blackred">

<stop offset="0%" stop-color="black" />

<stop offset="100%" stop-color="black" />

</linearGradient>

</defs>`;

hide(s);

});

simulation

.nodes(data.nodes)

.on("tick", ticked);

simulation.force("link")

.links(data.links);

function ticked() {

link

.attr("x1", function(d) { return d.source.x; })

.attr("y1", function(d) { return d.source.y; })

.attr("x2", function(d) { return d.target.x; })

.attr("y2", function(d) { return d.target.y; });

node

.attr("cx", function(d) { return d.x; })

.attr("cy", function(d) { return d.y; });

}

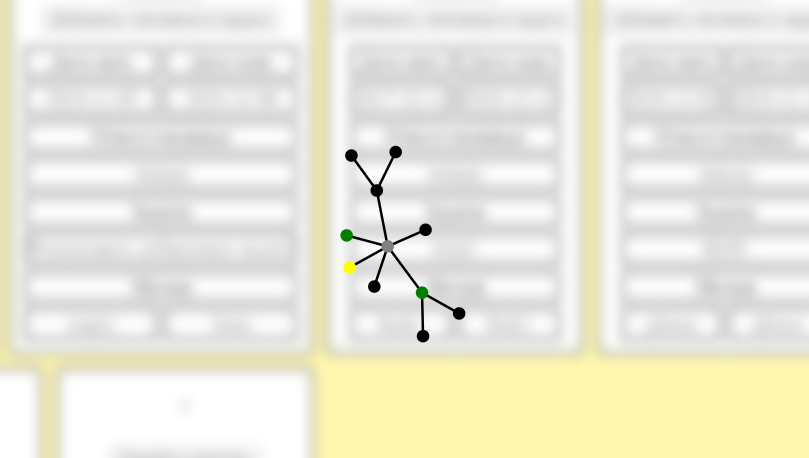
Данное меню при создании будет иметь следующий вид (рис. 3)

Рисунок 3 – «Шаблонная навигация на веб-странице»

# Тестирование

Валидация же данных происходит благодаря типу <input> и моему хранению данных, также при переводе в формат json данные получают правильный для postgresql вид, а ошибки в виде кавычек и других знаков, а также sql-инъекций решаются параметризацией запросов внутри самого запроса sql, а не стандартных параметров строки golang.

Для тестирования маршрутов я создал тестовые скрипты на языке golang и в таблице (табл. 1) будут указаны результаты с комментариями.

|  |  |  |  |
| --- | --- | --- | --- |
| Состав запроса | Полученный результат | Ожидаемый результат (Пусто, если совпало) | Комментарий |
| url := "http://localhost:8080/login"      method := "POST"      payload := strings.NewReader(`{          "login": "Admin",          "password": "Admin"      }`) | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MzEyNDcwMzUsInN1YiI6IkFkbWluIn0.5nxXHBnbCNAGh5KnaAo2yW3SCZ02-TD8bg\_zeuN3rnc |  |  |
| url = "http://localhost:8080/login"      method = "POST"      payload = strings.NewReader(`{          "login": "Admin",          "password": "Admiааn"      }`) | Отсутствие токена в ответе |  | Также ошибка на сайте |
| url = path\_2      method = "POST"      payload = strings.NewReader(`{              "login": "Admin",              "email": "Admin@gmail.com",              "fio": "Admin",              "password": "Admin"          }`) | {HAS} |  | Ответ когда происходит конфликт |
| url = "http://localhost:8080/registration"      method = "POST"      payload = strings.NewReader(`{              "login": "TEST1",              "email": "TEST@gmail.com",              "fio": "TEST TEST",              "password": "TEST1"          }`) | {OK} |  |  |
| url = "http://localhost:8080/selectByParent"      method = "POST"      payload = strings.NewReader(`{              "pi": 0          }`) | [{{1 2024-10-29T00:00:00Z 2024-12-06T00:00:00Z 0 Сделать Курсовую long term in process} [{Admin}]} {{6 2024-11-05T00:00:00Z 2025-12-12T00:00:00Z 0 1 0 0} [{Admin}]} {{13 2024-11-05T00:00:00Z 2024-11-06T00:00:00Z 0 Реализовать добавление задач2 major done} [{Admin}]} {{15 2017-12-12T00:00:00Z 2018-12-12T00:00:00Z 0 fu fu fu} [{Admin}]} {{17 2024-11-01T00:00:00Z 2024-11-15T00:00:00Z 0 KMS please please} [{Admin}]} {{14 2024-11-05T00:00:00Z 2024-11-07T00:00:00Z 0 fu1414141 fu fu} []} {{7 2024-11-05T00:00:00Z 2025-12-12T00:00:00Z 0 1-1 0 done} [{Admin}]} {{13 2024-11-05T00:00:00Z 2024-11-06T00:00:00Z 0 Реализовать добавление задач2 major done} [{Admin}]} {{7 2024-11-05T00:00:00Z 2025-12-12T00:00:00Z 0 1-1 0 done} [{Admin}]}] |  |  |
| url = "http://localhost:8080/selectByParent"      method = "POST"      payload = strings.NewReader(`{              "pi": null          }`) | Если значение является пустым ошибка выйдет не поддержке шифровке js, остальные просто выдадут ошибку bad request |  |  |
| url = "http://localhost:8080/children"      method = "POST"      payload = strings.NewReader(`{                  "id": 1              }`) | [{1 3 1 } {1 4 1 } {3 5 2 }] |  |  |
| url = "http://localhost:8080/children"      method = "GET" | [{0 1 1 in process} {0 6 1 0} {0 7 1 done} {0 13 1 done} {0 15 1 fu1} {0 17 1 please} {6 8 2 0} {6 9 2 0} {7 10 2 0} {7 11 2 0}] |  |  |
| url = "http://localhost:8080/createTask"      method = "POST"      payload = strings.NewReader(`{              "dateBeg": "2012-12-12",              "dateEnd": "2012-12-12",              "parentId": 0,              "task": "test",              "mark": "test",              "status": "test"          }`) | {OK} |  |  |
| url = "http://localhost:8080/createTask"      method = "POST"      payload = strings.NewReader(`{              "dateBeg": "2012-12-12",              "parentId": 0,              "task": "test",              "mark": "test",              "status": "test"          }`) | {ERROR}, т.к. отсутствует главный ключ |  | Любое отсутствие главного атрибута приведёт к ошибке, на сайте предусмотрены подобные ошибки, а также о неправильных данных |
| url = "http://localhost:8080/changeTask"      method = "POST"      payload = strings.NewReader(`{              "id": 1,              "dateBeg": "2012-12-12",              "dateEnd": "2012-12-12",              "parentId": 0,              "task": "test",              "mark": "test",              "status": "test"          }`) | {OK} |  | В запросе на изменение главное- иметь id задачи, если другие атрибуты отсутствую, или неверны, берутся старые |
| url = path\_7      method = "DELETE"      payload = strings.NewReader(`{              "pi": 1          }`) | {OK} |  |  |
| url = "http://localhost:8080/deleteTask"      method = "DELETE"      payload = strings.NewReader(`{              "pi": 100          }`) | {ERROR} |  | При указании несуществующего id, даётся ошибка, на сайте такой ошибки допустить невозможно, т.к. кнопки удаления привязаны к задачам |
| url = "http://localhost:8080/deleteTask"      method = "DELETE"      payload = strings.NewReader(`{              "pi": “1”          }`) | Bad Request |  | При несовпадении типов данных в всех ручных запросах будет подобная ошибка |
| url = "http://localhost:8080/deleteTask"      method = "PUT"      payload = strings.NewReader(`{              "pi": 1          }`) |  |  | Во всех маршрутах при неправильном методе просто не даётся никакой результат |
| url = "http://localhost:8080/task/addWorker"      method = "POST"      payload = strings.NewReader(`{              "idTask": 1,              "login": "User1"          }`) | {OK} |  |  |
| url = "http://localhost:8080/task/addWorker"      method = "POST"      payload = strings.NewReader(`{              "idTask": 1,              "login": "User1111"          }`) | {ERROR} |  | Ошибка при отсутствии указанного человека или задачи |
| url = "http://localhost:8080/report/left"      method = "POST"      payload = strings.NewReader(`{              "pi": 7          }`) | [{15 f1 f1 f1 365 -2157}] |  | Возвращает значение в порядке: id, текст, метка, статус, было дней, осталось дней, и сортирует по остатку |
| url = "http://localhost:8080/report/forLast"      method = "POST"      payload = strings.NewReader(`{              "pi": 100          }`) | [{13 Реализовать добавление задач2 major done 1}] |  |  |
| url = "http://localhost:8080/report/forLast"      method = "POST"      payload = strings.NewReader(`{              "pi": -100          }`) |  | Ошибка введённых данных | Ошибки не будет, как и результата |

Таблица 1 - «Тестирование маршрутов»

Тестирование функциональности элементов на сайте прошло все проверки.

# Заключение

По завершению проекта все задачи, что были поставлен были выполнены. У пользователя есть возможность создания, изменения просмотра задач в удобном интерфейсе, навигация задач представлена в удобном интерактивном окне, имеется возможность открытия доступа другим людям доступа к задаче и получение отчётов. Но с развитием проекта появлялись пути для расширения, на которые в силу их количества не было времени. Дальнейшее развитие для этого проекта подразумевало бы создание инструмента для фильтрации задач по любым параметрам, увеличение количества отчётов и создание более приятного шаблона, также создание более продвинутого стиля страницы, создание таблицы ресурсов задачи будь то файлы или параметр часов, денег.

При росте такого продукта вскоре также придёт необходимость асинхронного шифрования данных.

# Список использованных источников

1. Документация по веб-программированию [Электронный ресурс] // Documenting web technologies, including CSS, HTML, and JavaScript, since 2005. -Режим доступа: <https://developer.mozilla.org/en-US/docs/>
2. Примеры диаграмм [Электронный ресурс] // The best place to build, test, and discover front-end code.-Режим доступа: [Menu parent children as a diagram...](https://codepen.io/joellesenne/pen/KKzVMW)
3. JSON Web Token и Secure Sockets Layer [Электронный ресурс] // Хабр -Режим доступа: <https://habr.com/ru/articles/654191/>
4. Пакет для графиков [Электронный ресурс] The JavaScript library for bespoke data visualization -Режим доступа: <https://d3js.org>
5. Документация по веб-программированию [Электронный ресурс] // METANIT.COM Сайт о программировании -Режим доступа: <https://metanit.com/web/>
6. Документация по использованным языкам программирования [Электронный ресурс] // w3schools -Режим доступа: <https://www.w3schools.com/>
7. Свойства html элементов [Электронный ресурс] // HTML5 BOOK -Режим доступа: <https://html5book.ru/>
8. Статья про запросы с передачей json данных [Электронный ресурс] // Современный учебник JavaScript -Режим доступа: <https://learn.javascript.ru/fetch>
9. Статьи по JS [Электронный ресурс] // Современный учебник JavaScript -Режим доступа: <https://learn.javascript.ru/>
10. Документация по обращению БД с помощью golang [Электронный ресурс] // Querying for data -Режим доступа: <https://go.dev/doc/database/querying>
11. Как запускать SQL в Go с максимальным комфортом [Электронный ресурс] // Хабр -Режим доступа: <https://habr.com/ru/companies/vk/articles/692062/>
12. Документация по созданию http сервера и клиента на golang [Электронный ресурс] // http -Режим доступа: <https://pkg.go.dev/net/http>
13. Документация по http серверам в golang [Электронный ресурс] // Golang http - Create HTTPS Server and Client -Режим доступа: <https://www.golinuxcloud.com/golang-http/>
14. Исходные файлы и документация библиотек для создания Java Token на golang [Электронный ресурс] // GitHub -Режим доступа: <https://github.com/golang-jwt/jwt>
15. Golang JWT (Json Web Token) Examples [Электронный ресурс] // golinuxcloud -Режим доступа: <https://www.golinuxcloud.com/golang-jwt/>