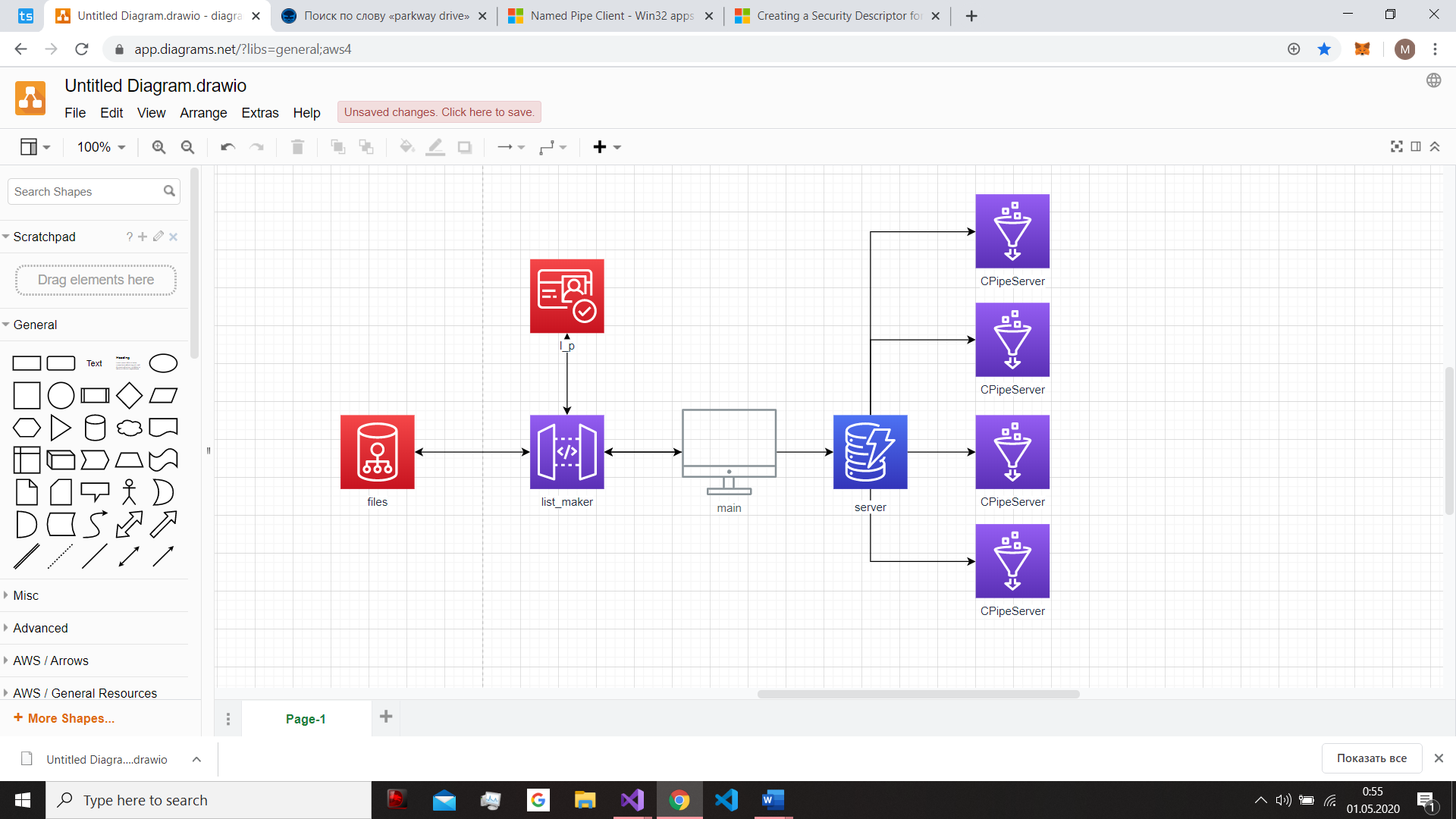
Содержание

[Server 2](#_Toc39436434)

[Struct task 2](#_Toc39436435)

[Class list\_maker 3](#_Toc39436436)

# Server



Сначала Main вызовет list\_maker и обработает файл с логинами и паролями.

Затем вызывается Server который занимается обслуживанием клиентов. Ему также передается список логинов и паролей созданный list\_maker.

Struct task{

Id; //индекс пайпа

Time;

}

В Server, после обработки состояния пайпа, клиент, если пароль неправильный, не будет отключен, а в task \_list будет добавлена задача. Проверка флажка, который определяет нужно ли ждать или можно отправить ответ будет добавлена в обработчик состояния пайпа. Обработка задач (проверка настало ли время отправки) будет вынесена в отдельный поток. У которого будет доступ к task \_list.

//занимается обработкой файла с логинами и паролями

Class list\_maker{

String path;

Unsigned max\_ password \_length;

Unsigned max\_login\_length;

list\_maker(string path){

this.path = path;

}

//получение списка с логинами и паролями, вызывается пользователем

Public: List <l\_p> Get\_list(){

String buf = read\_from\_file();

Get\_info(buf);

Make\_list(buf);

Return list;

}

//считываем файл

Private: String read\_from\_file(){

//считываем файл в строку и возвращаем данные

//throw если что пошло не так

}

//парсит одну строку

Private: l\_p get\_lp (string &str,){

//проверяем длину и корректность данных

//парсим строку, создаем l\_p, возвращаем

//throw если что пошло не так

}

//вычитывает и удаляет первую строку с данными о размере логина и пароля

Private: get\_info (string &buf){

//вычитываем первую строку

//заполняем поля max\_ password \_length, max\_login\_length

//удаляем первую строку

//throw если что пошло не так

}

//создание списка с l\_p;

Private: list<l\_p> Make\_list(string &buf){

list<l\_p> arr;

While{

//берем строки по очереди

//вызываем get\_lp();

//добавляем в arr

}

Return arr;

}

}

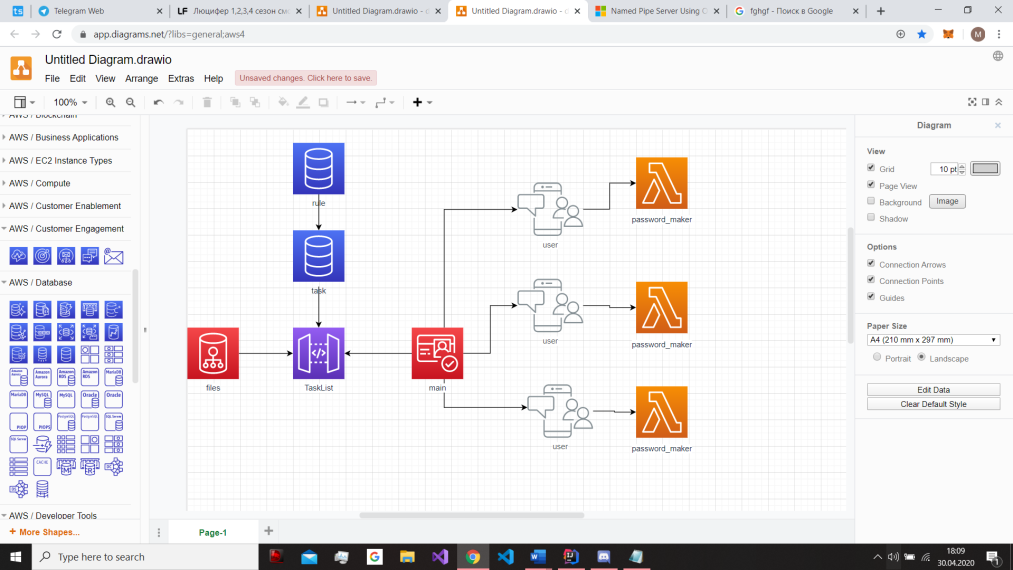
//логин + пароль

Struct l\_p{

List<string> Logins;

List<string> Passwords;

}



Структура файла с логинами и паролями выглядит следующим образом:

| Логин || Предположительный\_пароль || номер\_правила-опция1-опцияN |

Опции – это, например, диапазон символов, в котором будет происходить изменения в предложении, которые зависят от самого правила. Их может быть несколько.

Класс main вызывает TaskList, кидая ему путь к серверу и строку из списка для обработки, который в свою очередь вызывает Task, а он – rules. Далее вызывается User, который принимает обработанную строку и отправляет в PasswordMaker. User имеет связь с сервером.

Пользователь должен иметь возможность выбирать пароль с правилом, отправить его на обработку и пришедший массив паролей отправляет серверу (по строчно). Также пользователей может быть несколько штук.

//Класс, принимающий поделенную строку, кидает ее в PasswordMaker, принимает массив из паролей и отправляет его серверу

Class User{

Task[] tasks;

Pipe pipe;

User(){

this.tasks = TaskList. tasks;

this pipe = \*пайп\*

}

run(){

for (int p = 0; …){

for(auto i: PasswordMaker(tasks[i])){

//отправка на сервер…

}

}

}

}

//класс, создающий список задач, которые будут выполнены в дальнейшем.

Class TaskList{

public Task[] tasks = new Task;

String path;

//добавление пути, куда подключен клиент

TaskManager(str[] path){

this.path = path

}

//метод для добавления в массив поделенной строки из списка

tasks[] PathParse(String str) { // принимаем одну строку из списка

tasks.add(new Task(str)); // кидаем строку в Task и добавляем с массив

throw();

}

}

===========================================================================

//С учетом того, что может быть несколько опций, нужно их разделить по разделителю, для чего и нужен этот класс.

Struct rules(){

int id[];

int option[];

// запросить пришедшую сюда строку с правилами и опциями

// и потом возвращать массивы id[] и option[]

}

===========================================================================

//разделение строки и списка на отдельные компоненты (логин, пароль и команды)

Struct Task(){

str login;

str password;

rules qqqq[];

Task(String all){

//Делим строку через parse

this.login = Login;

//Присваиваем и остальным переменным этой структуры

}

}

//класс, который принимает один пароль и возвращает массив из видоизмененных паролей по правилу

Class PasswordMaker(){

TakePass(Task task){

string newPass[];

for (int I = 0; I <= ){

switch (taskID)

//case-ы, с перебором всех идентификаторов задач, тоесть просто ищем нужное правило, и вызываем нужную функцию.

}

return newPass[];

}

//методы, реализующие следующие правила:

1. Замена буков в определенном промежутке (пароль, ВекторБукв,промежуток)
2. Замена конкретной буквы (пароль, ВекторБукв)
3. Перевод (пароль)
4. Смена регистра букв (пароль)
5. Добавление символа в определенном промежутке (пароль, ВекторБукв, промежуток)
6. грубаяСила()

}