**СОДЕРЖАНИЕ**

[Введение 3](#_Toc132049495)

[1 Общие сведенья об организации 4](#_Toc132049496)

[1.1 История предприятия 4](#_Toc132049497)

[1.2 Информация об организации 5](#_Toc132049498)

[1.3 Виды выпускаем продукции](#_Toc132049500) 6

[2 Основные методолгии разработки ПО и их защита](#_Toc132049501) 8

[2.1 Основные модели разработки ПО](#_Toc132049502) 8

[2.2 Способы защиты информации](#_Toc132049503) 10

[3 Описание используемого ПО на предприятие](#_Toc132049505) 13

[3.1 Разработка сопроводительной документации](#_Toc132049506) 13

[3.2 Разработка программного продукта](#_Toc132049507) 15

[4 Анализ требований ЕСПД, ГОСТ, ОСТ, ISO для оформления программной документации](#_Toc132049508) 19

[5 Анализ технического задания и графической документации 2](#_Toc132049509)6

[5.1 Техническое задание на один из видов выпускаемой продукции 2](#_Toc132049510)6

[5.2 Графическая документация](#_Toc132049511) 27

[Заключение](#_Toc132049513) [3](#_Toc132049512)0

Список использованных источников [3](#_Toc132049514)1

[Приложение](#_Toc132049515) А (обязательное) [Листинг программы 3](#_Toc132049516)2

**ВВЕДЕНИЕ**

На данный момент производственная практика является важным этапом в жизни студентов и их дальнейшего карьерного роста так как производственная практика позволяет получить практические навыки работы в команде, изучить рабочее пространство изнутри и изучить весь процесс разработки ПО.

Во время производственной практики у студентов есть возможность ознакомиться с реальными проектами, а также получить опыт разработки приложения под руководством опытных руководителей.

Главным этапом производственной практики является знакомство с компанией. Анализируются основные сферы деятельности и задачи персонала, занятого в определенной области.

Практика пройдена на ООО «Cакцесс». Данная компания специализируется на разработке приложений и продажи их на внешний рынок. Процесс разработки приложений зависит от команды разработки и типа разрабатываемого приложения следовательно компания имеет как множество специалистов узкого профиля так и специалистов широкого профиля.

Во время нахождения на производственной практике были определенны следующие целы и задачи:

− изучить историю предприятия;

− изучить вопросы по охране труда и технике безопасности;

− изучить процессы разработки приложений и задачи специалистов данного профиля;

− приобрести практический опыт разработки приложений;

− выполнить практической задание.

Индивидуальным задание от компании являться разработка LWC (Lightning Web Component) для запуска расписания по средством ввода специального акронима типа СRON.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

[1] Каскадная модель (waterfall model): [Электронный ресурс] - Режим доступа: https://gb.ru/posts/waterfall (Дата доступа 05.06.23)

[2] Итеративная и инкрементальная модель: [Электронный ресурс] - Режим доступа: https://evergreens.com.ua/ru/articles/software-development-metodologies.html (Дата доступа 05.06.23)

# [3] Схема Agile (Scrum) [Электронный ресурс]. - Режим доступа: https://www.scrum.org/resources/what-scrum-module (Дата доступа 24.07.23)

[4] Уровни Доступа в Salesforce [Электронный ресурс]. - Режим доступа: https://trailhead.salesforce.com/content/learn/modules/data\_security/data\_security\_overview (Дата доступа 12.06.23)

[5] Уровни Доступа организация [Электронный ресурс]. - Режим доступа: https://trailhead.salesforce.com/content/learn/modules/data\_security/data\_security\_org (Дата доступа 12.06.23)

[6] Уровни Доступа объект [Электронный ресурс]. - Режим доступа: https://trailhead.salesforce.com/content/learn/modules/data\_security/data\_security\_objects (Дата доступа 12.06.23)

[7] Уровни Доступа поле [Электронный ресурс]. - Режим доступа: https://trailhead.salesforce.com/content/learn/modules/data\_security/data\_security\_fields (Дата доступа 12.06.23)

[8] Уровни Доступа запись [Электронный ресурс]. - Режим доступа: https://trailhead.salesforce.com/content/learn/modules/data\_security/data\_security\_records (Дата доступа 12.06.23)

[9] Salesforce Inspector [Электронный ресурс]. - Режим доступа: https://developer.salesforce.com/docs/atlas.en−us.lightning.meta/lightning/inspector\_intro.html (Дата доступа 17.06.23)

[10] ГОСТ 19.103–77 [Электронный ресурс]. - Режим доступа: https://internet-law.ru/gosts/gost/33235/ (Дата доступа 20.06.23)

[11] ГОСТ 19.106–78 [Электронный ресурс]. - Режим доступа: https://internet-law.ru/gosts/gost/32437/ (Дата доступа 20.06.23)

[12] ISO/IEC/IEEE 29148:2018 [Электронный ресурс]. - Режим доступа: https://www.iso.org/standard/72089.html (Дата доступа 20.06.23)

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Листинг программы**

import { LightningElement, track } from "lwc";

import getCurrentlyScheduledCron from "@salesforce/apex/LWCSchedulingService.getCurrentlyScheduleCron";

import runFirstJob from "@salesforce/apex/LWCSchedulingService.runFirstJob";

import checkFirstJobStatus from "@salesforce/apex/LWCSchedulingService.checkFirstJobStatus";

import scheduleJob from "@salesforce/apex/LWCSchedulingService.scheduleJob";

import deleteScheduledJob from "@salesforce/apex/LWCSchedulingService.deleteScheduledJob";

export default class LwcScheduler extends LightningElement {

cronJobName = "Create Daily Account Record";

methodName = "createAccountRecord";

@track currentCronAsTime;

currentCronAsString;

state; // test, schedule, reschedule

loading;

connectedCallback() {

this.loading = true;

this.getScheduledCron();

}

/\*\*

\* On component load - we want to check to see if the job is currently scheduled. If it is

\* scheduled - we can modify the state appropriatley.

\*/

getScheduledCron() {

getCurrentlyScheduledCron({ cronJobName: this.cronJobName })

.then(result => {

switch (result) {

case "test":

this.state = result;

break;

case "schedule":

this.state = result;

break;

default:

this.currentCronAsTime = this.convertCronToTime(result);

console.log("Job Scheduled for: " + this.currentCronAsTime);

this.state = "reschedule";

}

this.stopLoading(500);

})

.catch(error => {

this.stopLoading(500);

});

}

convertCronToTime(result) {

let cronArray = result.split(" ");

let [second, minute, hour] = cronArray;

return `${hour}:${minute}:00.000`;

}

runFirstJob() {

this.loading = true;

runFirstJob({})

.then(data => {

this.checkFirstSecurityJobStatus();

})

.catch(error => {

this.stopLoading(500);

});

}

checkFirstJobStatus() {

checkFirstJobStatus({

submittedDatetime: this.dateTimeSubmitted,

methodName: this.methodName

})

.then(result => {

switch (result) {

case "Completed":

this.state = "schedule";

this.stopLoading;

break;

case ("Aborted", "Failed"):

this.stopLoading(500);

console.log(data.ExtendedStatus);

default:

setTimeout(() => {

console.log("Checking");

this.checkFirstJobStatus();

}, 100);

}

})

.catch(error => {

console.log(error.message);

});

}

scheduleApexJob() {

this.loading = true;

scheduleJob({

cronString: this.currentCronAsString,

cronJobName: this.cronJobName

})

.then(data => {

console.log(data);

if (data) {

this.state = "reschedule";

this.getScheduledCron();

} else {

this.stopLoading(500);

console.log("Unable to Schedule Job");

}

})

.catch(error => {

this.stopLoading(500);

console.log(error.message);

});

}

deleteJob() {

this.loading = true;

deleteScheduledJob({ cronJobName: this.cronJobName })

.then(data => {

console.log(data);

if (data) {

this.state = "schedule";

this.currentCronAsTime = "";

this.stopLoading(500);

console.log("Job Deleted");

} else {

this.stopLoading(100);

console.log("we were unable to delete this job");

}

})

.catch(error => {

this.stopLoading(100);

console.log(error.message);

});

}

handleTimeChange(event) {

let time = event.target.value;

let [hour, minute, seconds] = time.split(":");

this.currentCronAsString = `0 ${minute} ${hour} ? \* \* \*`;

}

/\*\*

\* The stopLoading utility is used to control a consistant state experience for the user - it ensures that

\* we don't have a flickering spinner effect when the state is in flux.

\* @param {timeoutValue} timeoutValue

\*/

stopLoading(timeoutValue) {

setTimeout(() => {

this.loading = false;

}, timeoutValue);

}

}

public with sharing class LWCSchedulingService {

/\*\*

\* The setup utilities are used to manage scheduled jobs throughout the app. With these you can see the details

\* of a schedule job, schedule a job, or delete a scheduled job.

\*/

@AuraEnabled

public static void runFirstJob() {

LWCSchedulerDemo.createAccountRecord();

}

@AuraEnabled

public static Datetime getCurrentDateTime() {

return Datetime.now();

}

@AuraEnabled

public static AsyncApexJob checkFirstJobStatus(

Datetime submittedDatetime,

String methodName

) {

List<AsyncApexJob> asyncApexJobs = [

SELECT Id, Status, ExtendedStatus

FROM AsyncApexJob

WHERE MethodName = :methodName AND CompletedDate >= :submittedDatetime

WITH SECURITY\_ENFORCED

LIMIT 1

];

AsyncApexJob asyncApexJob = (asyncApexJobs.size() == 1)

? asyncApexJobs.get(0)

: null;

if (asyncApexJob != null) {

return asyncApexJob;

}

return null;

}

@AuraEnabled

public static String getCurrentlyScheduleCron(String cronJobName) {

List<CronTrigger> cronTriggers = [

SELECT Id, CronExpression

FROM CronTrigger

WHERE CronJobDetail.Name = :cronJobName

WITH SECURITY\_ENFORCED

LIMIT 1

];

CronTrigger cronTrigger = (cronTriggers.size() == 1)

? cronTriggers.get(0)

: null;

if (cronTrigger != null) {

return cronTrigger.CronExpression;

}

return null;

}

@AuraEnabled

public static Boolean scheduleJob(String cronString, String cronJobName) {

try {

LWCScheduler scheduler = new LWCScheduler();

System.schedule(cronJobName, cronString, scheduler);

return true;

} catch (Exception e) {

System.debug(e.getMessage());

return false;

}

}

@AuraEnabled

public static Boolean deleteScheduledJob(String cronJobName) {

try {

List<CronTrigger> cronTriggers = [

SELECT Id, CronExpression

FROM CronTrigger

WHERE CronJobDetail.Name = :cronJobName

WITH SECURITY\_ENFORCED

LIMIT 1

];

CronTrigger cronTrigger = (cronTriggers.size() == 1)

? cronTriggers.get(0)

: null;

System.abortJob(cronTrigger.Id);

return true;

} catch (Exception e) {

System.debug(e.getMessage());

return false;

}

}

}