ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Профессор департамента программной инженерии факультета компьютерных наук, к.т.н

_____ Е. М. Гринкруг
«_____» _____ 2020 г.

УТВЕРЖДАЮ
Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук

_____ В. В. Шилов
«_____» _____ 2020 г.

Инв. № подл | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата

СИСТЕМА УПРАВЛЕНИЯ ЗАДАНИЯМИ ПО АВТОМАТИЧЕСКОМУ СБОРУ ДАННЫХ ИЗ СЕТИ ИНТЕРНЕТ

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.13-01 34 01-1

Исполнитель: студент группы БПИ 174
_____ Д. Ю. Редникина
«_____» _____ 2020 г.

Москва 2020

УТВЕРЖДЕН
RU.17701729.04.13-01 34 01-1

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

СИСТЕМА УПРАВЛЕНИЯ ЗАДАНИЯМИ ПО АВТОМАТИЧЕСКОМУ СБОРУ ДАННЫХ ИЗ СЕТИ ИНТЕРНЕТ

Руководство программиста

RU.17701729.04.13-01 34 01-1

Листов 28

Москва 2020

Содержание

1 Назначение программы **4**
1.1 Функциональное назначение 4
1.2 Эксплуатационное назначение 4
1.3 Состав выполняемых функций 4

2 Обращение к сервису **7**
2.1 API - список 7
2.1.1 Authorization 8
2.1.2 Registration 9
2.1.3 Project 10
2.1.4 Membership 13
2.1.5 Crawlers 16
2.1.6 Onetime jobs 17
2.1.7 Periodic Jobs 21

Приложение А **26**

Список источников **27**

Лист регистрации изменений **28**

1 Назначение программы

1.1 Функциональное назначение

Система будет применяться как средство управления проектами по созданию, редактированию и запуску веб краулеров для сбора данных в сети интернет. Продукт позволит следить за запусками в режиме реального времени, а также создавать периодические запуски по расписанию.

1.2 Эксплуатационное назначение

Программа будет использоваться как инструмент для самостоятельной или совместной работы над проектами для запуска, управления сбора данных с помощью веб краулеров в сети интернет.

Таким образом, программный продукт позволит создавать, запускать образы пауков (см. 2.1.7) для сбора, управления, логирования и дальнейшего экспорта данных в целях сбора, изучения и мониторинга данных (см. 2.1.7).

1.3 Состав выполняемых функций

Следующие требования зафиксированы в документе «Система управления заданиями по автоматическому сбору данных из сети Интернет. Техническое задание» к составу выполняемых функций:

1. **Авторизация**
   Чтобы использовать сервис, клиентская программа должна иметь возможность авторизоваться в системе с помощью REST API
   (a) Для регистрации пользователю нужно указать следующие данные
      i. Почта - уникальна для каждого зарегистрированного пользователя;
      ii. Имя - длина больше 1 символ;
      iii. Логин - длина больше 2 символов;
      iv. Пароль - длина больше 2 символов;
   (b) Для авторизации пользователя в системе должны быть указаны следующие данные
      i. Почта;
      ii. Пароль;
2. **Проекты**
   Должны быть реализованы запросы REST API для предоставления клиенту следующей функциональности
   (a) Создание проекта со следующей информацией
      i. Имя проекта;
      ii. Описание проекта - опциональное поле;
   (b) Обновление метаданных о проекте (редактирование) могут быть обновлены только участником с минимальным уровнем дотупа `ReadAndWrite`. Следующие данные могут быть обновлены:

i. Имя проекта;
ii. Описание проекта;
iii. Настройки проекта для запуска краулеров;
iv. Аргументы для запуска краулеров проекта;

(c) Обновление `egg` файла проекта (редактирование) – минимальный уровень доступа участника, обновляющий данные о проекте `ReadAndWrite`.

(d) Удаление данных о проекте. Удалить проект может только владелец `Owner`.

(e) Просмотр списка проектов (с пагинацией), к которым у пользователя есть как минимум `ReadOnly` доступ.

3. **Участники проектов**
Должны быть реализованы запросы REST API для предоставления клиенту следующей функциональности

(a) Просмотр информации об участниках проекта;
i. Имя, почта, логин участника;
ii. Статус участника в проекте (`ReadOnly`, `ReadAndWrite` или `Owner`);

(b) Обновление статуса участника проекта. Это действие совершать может только владелец проекта;

(c) Удаление участника из проекта. Данное действие может совершать только владелец проекта;

(d) Добавление нового участника с указанными правами на редактирование. Данное действие может совершать только владелец проекта;

4. **Краулеры**
Должны быть реализованы запросы REST API для предоставления клиенту следующей функциональности

(a) Просмотр списка краулеров проекта;

(b) Редактирование информации о краулере для последующих запусков. Следующая информация может быть изменена
i. Настройки краулера для запуска;
ii. Аргументы для запуска;

5. **Запуски краулеров**
Должны быть реализованы запросы REST API для предоставления клиенту следующей функциональности

(a) Просмотр списка запусков в определенном статусе (`Pending`, `Running` или `Finished`) с пагинацией, совершенных в проектах, к которым у пользователя есть как минимум `ReadOnly` доступ;

(b) Редактирование запуска - остановка запуска, перевод его в состояние `Finished`. Операция может быть применена только к запускам в состоянии `Running` или `Pending`;

(c) Удаление запуска - удаление всех данных о запуске из базы данных. Операция может быть применена только к запускам в состоянии `Finished`;

(d) Создание запуска со следующей информацией

i. Краулер, с которым происходит запуск;
ii. Настройки запуска – это могут быть как и предопределенные настроки на `scrapyd` [1], так и собственные настройки;
iii. Аргументы запуска – аргументы для запуска краулера, которые передаются через командную строку;
iv. Описание запуска;

6. **Периодические запуски**
Должны быть реализованы запросы REST API для предоставления клиенту следующей функциональности

(a) Просмотр списка периодических запусков с пагинацией;
(b) Редактирование следующей информации о периодическом запуске
i. Настройки будущих запусков – это могут быть как и предопределенные настроки на `scrapyd`, так и собственные настройки;
ii. Аргументы будущих запусков – аргументы для запуска краулера, которые передаются через командную строку;
iii. Краулер, с помощью которого будет совершен запуск;
iv. cron-expression расписания запуска;
(c) Удаление периодического запуска;
(d) Отмена последующих запусков - перевод периодической задачи в состояние `Disabled`;
(e) Возобновление запусков - перевод периодической задачи в состояние `Enabled`;
(f) Создание периодического запуска со следующими данными
i. Название;
ii. Описание – опциональное;
iii. Краулер;
iv. Приоритетность, влияющая на очередь запусков (`Low`, `Normal` или `High`);
v. Статус (`Enabled` или `Disabled`);
vi. Настройки будущих запусков – это могут быть как и предопределенные настроки на `scrapyd`, так и собственные настройки;
vii. Аргументы будущих запусков – аргументы для запуска краулера, которые передаются через командную строку;

[1] `http://doc.scrapy.org/en/latest/topics/settings.html`

2 Обращение к сервису

К данному сервису обращение осуществляется посредством REST запросов [?]. Далее в этом разделе приведены все возможные запросы к сервису с примерами аргументов к нему.

2.1 API - список

Crawlers

- `get /api/projects/{projectId}/crawlers`
- `put /api/projects/{projectId}/crawlers/{crawlerId}`

Login

- `post /api/auth/signin`

Logout

- `get /api/auth/logout`

Membership

- `put /api/projects/{projectId}/memberships/{guestId}/{guestAccess}`
- `delete /api/projects/{projectId}/memberships/{guestId}`
- `get /api/projects/{projectId}/memberships`

OnetimeJobs

- `put /api/projects/{projectId}/jobs/{jobScrapydId}/{jobId}`
- `delete /api/projects/{projectId}/jobs/{jobScrapydId}/{jobId}`
- `get /api/projects/jobs/{limit}/{status}`
- `post /api/projects/{projectId}/jobs`

PeriodicJobs

- `post /api/projects/{projectId}/periodicJobs`
- `put /api/projects/{projectId}/periodicJobs/{periodicJobId}`
- `delete /api/projects/{projectId}/periodicJobs/{periodicJobId}`
- `put /api/projects/{projectId}/periodicJobs/{periodicJobId}/disable`
- `put /api/projects/{projectId}/periodicJobs/{periodicJobId}/enable`
- `get /api/projects/{projectId}/periodicJobs/{limit}`

Projects

- `post /api/projects`
- `delete /api/projects/{projectId}`
- `put /api/projects/{projectId}/deploy`
- `get /api/projects/{limit}`
- `put /api/projects/{projectId}`

Registration

- `post /api/auth/signup`

2.1.1 Authorization

`post /api/auth/signin`

Get authentication token (signIn)

Request body

body SignIn (required)
Body Parameter — Credentials

Return type

Cookie

Example data

Content-Type: application/json

```
{
  "cookie" : "cookie",
  "login" : "login"
}
```

Responses

200 successful operation Cookie

400 SignInBadRequest

403 InvalidCredentialsProvided

`get /api/auth/logout`

Logout (logout)

Return type

ActionAnyContent

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionAnyContent

401 Unauthorized

2.1.2 Registration

`post /api/auth/signup`

Get authentication token (signUp)

Request body

body SignUp (required)
Body Parameter — Credentials

Return type

Cookie

Example data

Content-Type: application/json

```
{
  "cookie" : "cookie",
  "login" : "login"
}
```

Responses

200 successful operation Cookie

400 SignUp body bad request

403 EmailWrongFormat

409 UserAlreadyExistsMessage

2.1.3 Project

`post /api/projects`

Create project (createProject)

Request body

body ProjectForm (required)
Body Parameter — Form with initial project data

Return type

UUID

Example data

Content-Type: application/json

```
"046b6c7f-0b8a-43b9-b35d-6489e6daee91"
```

Responses

200 successful operation UUID

400 ProjectFormBadRequest

401 Unauthorized

500 Couldn't create projects

`delete /api/projects/{projectId}`

Delete project (deleteProject)
We can't delete project from our DB in case we encounter error deleting it from scrapyd

Path parameters

projectId (required)
Path Parameter — format: int64

Return type

ActionAnyContent

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionAnyContent

401 Unauthorized

403 Do not have permission to delete project

422 Error deleting project from scrapyd

`put /api/projects/{projectId}/deploy`

Deploy project's eggfile to scrapyd (deployProject)

Path parameters

projectId (required)
Path Parameter — format: int64

Form parameters

eggFile (required)
Form Parameter —

Return type

array[Crawler]

Example data

Content-Type: application/json

```
[ {
  "settings" : { },
  "name" : "name",
  "id" : 0,
  "projectId" : 6
}, {
  "settings" : { },
  "name" : "name",
  "id" : 0,
  "projectId" : 6
} ]
```

Responses

200 successful operation

401 Unauthorized

403 Do not have permission to deploy project

422 Error getting eggfile from multipart form/data or error deploy to scrapyd

---

`get /api/projects/{limit}`

Get projects (getProjects)

Path parameters

limit (required)
Path Parameter — Limit for request format: int32

Query parameters

id (optional)
Query Parameter — ID excludeFrom format: int64

Return type

array[Project]

Example data

Content-Type: application/json


```
[ {
  "createdAt" : 6,
  "eggfile" : [ "eggfile", "eggfile" ],
  "spidersSettings" : { },
  "changedBy" : "046b6c7f-0b8a-43b9-b35d-6489e6daee91",
  "name" : "name",
  "description" : "description",
  "changedAt" : 1,
  "id" : 0,
  "ownerId" : "046b6c7f-0b8a-43b9-b35d-6489e6daee91"
}, {
  "createdAt" : 6,
  "eggfile" : [ "eggfile", "eggfile" ],
  "spidersSettings" : { },
  "changedBy" : "046b6c7f-0b8a-43b9-b35d-6489e6daee91",
  "name" : "name",
  "description" : "description",
  "changedAt" : 1,
  "id" : 0,
  "ownerId" : "046b6c7f-0b8a-43b9-b35d-6489e6daee91"
} ]
```

Responses

200 successful operation

401 Unauthorized

500 Couldn't get projects

`put /api/projects/{projectId}`

Change project metadata (updateProjectMetadata)

Path parameters

projectId (required)
Path Parameter — format: int64

Request body

body ProjectChangeForm (required)
Body Parameter — Form with metadata to be changed

Return type

ActionProjectChangeForm

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionProjectChangeForm

401 Unauthorized

403 Do not have permission to change project

2.1.4 Membership

`put /api/projects/{projectId}/memberships/{guestId}/{guestAccess}`

Add or change participant of project (addParticipants)
Insert or update

Path parameters

projectId (required)
Path Parameter — format: int64
guestId (required)
Path Parameter — format: uuid
guestAccess (required)
Path Parameter —

Return type

ActionAnyContent

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionAnyContent

401 Unauthorized

403 Dont have permission to specified project

`delete /api/projects/{projectId}/memberships/{guestId}`

Delete user from membership list (deleteParticipant)
Only owner can delete from membership list

Path parameters

projectId (required)
Path Parameter — format: int64
guestId (required)
Path Parameter — format: uuid

Return type

ActionAnyContent

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionAnyContent

401 Unauthorized

403 Do not have permission to delete member

`get /api/projects/{projectId}/memberships`

Get list of members for project (getParticipants)
Not paginated

Path parameters

projectId (required)
Path Parameter — format: int64

Return type

array[Member]

Example data

Content-Type: application/json

```
[ {
  "accessRight" : { },
  "user" : {
    "name" : "name",
    "id" : "046b6c7f-0b8a-43b9-b35d-6489e6daee91",
    "login" : "login",
    "email" : "email"
  }
}, {
  "accessRight" : { },
  "user" : {
    "name" : "name",
    "id" : "046b6c7f-0b8a-43b9-b35d-6489e6daee91",
    "login" : "login",
    "email" : "email"
  }
} ]
```

Responses

200 successful operation

401 Unauthorized

403 Dont have permission to specified project

500 Couldn't get list of members

2.1.5 Crawlers

`get /api/projects/{projectId}/crawlers`

List spiders (listSpiders)
List spiders of project without pagination

Path parameters

projectId (required)
Path Parameter — Project ID format: int64

Query parameters

version (optional)
Query Parameter — Version of the project

Return type

array[Crawler]

Example data

Content-Type: application/json

```
[ {
  "settings" : { },
  "name" : "name",
  "id" : 0,
  "projectId" : 6
}, {
  "settings" : { },
  "name" : "name",
  "id" : 0,
  "projectId" : 6
} ]
```

Responses

200 successful operation

401 Unauthorized

403 Couldn't get project's spiders due to access rights permission

422 Coulnd't get spiders from DB

`put /api/projects/{projectId}/crawlers/{crawlerId}`

Update spider's settings (updateSpider)
Updates only spider's settings. `projectId` should match `spiderId`

Path parameters

projectId (required)
Path Parameter — format: int64
crawlerId (required)
Path Parameter — format: int64

Request body

body SpiderChangeForm (required)
Body Parameter — Form with new settings

Return type

ActionSpiderChangeForm

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionSpiderChangeForm

400 Bad format SpiderChangeForm

401 Unauthorized

403 Can't change spider's data due to access right permission

---

2.1.6 Onetime jobs

`put /api/projects/{projectId}/jobs/{jobScrapydId}/{jobId}`

Cancel running and pending tasks (cancel)
It moves both of the statuses to finished

Path parameters

projectId (required)
Path Parameter — format: int64
jobScrapydId (required)
Path Parameter — format: uuid
jobId (required)
Path Parameter — format: int64

Return type

UUID

Example data

Content-Type: application/json

```
"046b6c7f-0b8a-43b9-b35d-6489e6daee91"
```

Responses

200 successful operation UUID

401 Unauthorized

403 NoAccess

422 WrongStatus

`delete /api/projects/{projectId}/jobs/{jobScrapydId}/{jobId}`

Deletes finished job execution instance (deleteJob)
It removes all the information from DB

Path parameters

projectId (required)
Path Parameter — format: int64
jobScrapydId (required)
Path Parameter — format: uuid
jobId (required)
Path Parameter — format: int64

Return type

UUID

Example data

Content-Type: application/json

```
"046b6c7f-0b8a-43b9-b35d-6489e6daee91"
```

Responses

200 successful operation UUID

401 Unauthorized

403 User doesn't have at least ReadAndWrite access

422 Couldn't delete job execution

`get /api/projects/jobs/{limit}/{status}`

Get list of job executions with pagination (getJobsExecutions)

`With pagination. Get all of the current jobs for all user's project.`

Path parameters

limit (required)
Path Parameter — Limit for request format: int32
status (required)
Path Parameter — Status of job

Query parameters

fromId (optional)
Query Parameter — ID excludeFrom format: int64

Return type

array[Job]

Example data

Content-Type: application/json

```
[ {
  "jobInstanceId" : 6,
  "scrapydId" : "046b6c7f-0b8a-43b9-b35d-6489e6daee91",
  "project" : {
    "name" : "name",
    "id" : 2
  },
  "startTime" : 1,
```

```
  "id" : 0,
  "endTime" : 5,
  "spider" : {
    "name" : "name",
    "id" : 5
  },
  "status" : { }
}, {
  "jobInstanceId" : 6,
  "scrapydId" : "046b6c7f-0b8a-43b9-b35d-6489e6daee91",
  "project" : {
    "name" : "name",
    "id" : 2
  },
  "startTime" : 1,
  "id" : 0,
  "endTime" : 5,
  "spider" : {
    "name" : "name",
    "id" : 5
  },
  "status" : { }
} ]
```


Responses

200 successful operation

401 Unauthorized

500 Couldn't get jobs

`post /api/projects/{projectId}/jobs`

Schedule onetime job (schedule)
User has to have `ReadAndWrite` access to project. Initial status of the job = `pending`.Creates and starts new job with chosen crawler

Path parameters

projectId (required)
Path Parameter — format: int64

Request body

body OnetimeJobForm (required)
Body Parameter — Form with settings of onetime job for scheduling

Return type

ActionOnetimeJobForm

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionOnetimeJobForm

400 Bad format OnetimeJobForm

401 Unauthorized

403 Can't schedule job due to access right permission

409 Couldn't schedule job

2.1.7 Periodic Jobs

`post /api/projects/{projectId}/periodicJobs`

Creates periodic job (jobInstance) (addPeriodicJob)
Creates JonInstance in DB. Schedules jobs according to specified `cron expression`. Checks for user access rights.

Path parameters

projectId (required)
Path Parameter — format: int64

Request body

body PeriodicJobCreateForm (required)
Body Parameter — Form to create periodic job

Return type

UUID

Example data

Content-Type: application/json

```
"046b6c7f-0b8a-43b9-b35d-6489e6daee91"
```

Responses

200 successful operation UUID

401 Unauthorized

403 Dont have at least write access to specified project

422 Couldn't create job instance

`put /api/projects/{projectId}/periodicJobs/{periodicJobId}`

Changes the periodic Job data (changePeriodicJob)
Checks for user access rights and job-project connection.

Path parameters

projectId (required)
Path Parameter — format: int64
periodicJobId (required)
Path Parameter — format: int64

Request body

body PeriodicJobChangeForm (required)
Body Parameter — Form to change periodic job data

Return type

ActionPeriodicJobChangeForm

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionPeriodicJobChangeForm

401 Unauthorized

403 Dont have at least write access to specified project or job-project don't correspond

422 Couldn't change job instance

500 Error performing the update in DB

`delete /api/projects/{projectId}/periodicJobs/{periodicJobId}`

Delete periodic job (deletePeriodicJob)
Deletes periodic job instance (changed type to Onetime) and cancels all of the future job scheduled.

Path parameters

projectId (required)
Path Parameter — format: int64
periodicJobId (required)
Path Parameter — format: int64

Return type

ActionAnyContent

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionAnyContent

401 Unauthorized

403 NoPermission

422 JobCouldn'tBeDeleted

`put /api/projects/{projectId}/periodicJobs/{periodicJobId}/disable`

Sets status of periodicJob to disabled. (disable)
Cancels all of the future job scheduled. Does not modify running type, only running status.

Path parameters

projectId (required)
Path Parameter — format: int64
periodicJobId (required)
Path Parameter — format: int64

Return type

ActionAnyContent

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionAnyContent

`put /api/projects/{projectId}/periodicJobs/{periodicJobId}/enable`

Enable scheduling jobs. (enable)
Continues to schedule job executions. Does not modify running type, only running status.

Path parameters

projectId (required)
Path Parameter — format: int64
periodicJobId (required)
Path Parameter — format: int64

Return type

ActionAnyContent

Example data

Content-Type: application/json

```
{ }
```

Responses

200 successful operation ActionAnyContent

`get /api/projects/{projectId}/periodicJobs/{limit}`

Get list of periodic jobs with pagination (getPeriodicJobs)
Gets data from DB. No requests to scrapyd needed. User has to have access (at least `readonly`) to requested project

Path parameters

projectId (required)
Path Parameter — format: int64
limit (required)
Path Parameter — format: int32

Query parameters

exclusiveFrom (optional)
Query Parameter — format: int64

Return type

array[JobInstance]

Example data

Content-Type: application/json


```
[ {
  "cron" : "cron",
  "settings" : { },
  "description" : "description",
  "id" : 0,
  "title" : "title",
  "priority" : { },
  "projectId" : 6,
  "spider" : 1
}, {
  "cron" : "cron",
  "settings" : { },
  "description" : "description",
  "id" : 0,
  "title" : "title",
  "priority" : { },
  "projectId" : 6,
  "spider" : 1
} ]
```


Responses

200 successful operation

401 Unauthorized

403 Dont have at least read access to specified project

500 Couldn't get list of periodic jobs

ПРИЛОЖЕНИЕ А

Используемые понятия и определения

Web scraping – это сбор данных с различных интернет-ресурсов. Общий принцип его работы можно объяснить следующим образом: некий автоматизированный код выполняет GET-запросы на целевой сайт и получая ответ, парсит HTML-документ, ищет данные и преобразует их в заданный формат.

Проект – сущность для объединения и предоставления доступа к запускам/краулерам/периодическим задачам.

Веб краулер – программа, являющаяся составной частью поисковой системы и предназначенная для перебора страниц Интернета с целью занесения информации о них в базу данных поисковика. Неотъемлемая часть проекта. Именно с помощью пауков пользователь может “краулить” сайты для сбора необходимой информации.

Запуск – единоразовый запуск краулера с настройками и аргументами, указанными для этого запуска.

Периодический запуск – запуск с множеством настроек, повторяющийся в определенные периоды времени (запуски по cron-expression).

Список источников

[1] Github scrapyd/scrapyd [Электронный ресурс] URL: `https://github.com/scrapy/scrapyd` (Дата обращения: 16.04.2020, режим доступа: свободный)

[2] Единая система программной документации – М.: ИПК, Издательство стандартов, 2000, 125 стр.

[3] Postgresql [Электронный ресурс] URL:`https://www.postgresql.org` (Дата обращения: 16.04.2020, режим доступа: свободный)

[4] Play-framework [Электронный ресурс] URL:`https://www.playframework.com` (Дата обращения: 16.04.2020, режим доступа: свободный)

[5] Slick [Электронный ресурс] URL: `https://scala-slick.org`(Дата обращения: 16.04.2020, режим доступа: свободный)

Лист регистрации изменений									
Изм.	Номера листов				Всего листов в документе	№ документа	Входящий № сопроводит. докум. и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					