

**ЦИФРОВАЯ ПЛАТФОРМА «АВАНГАРД»**

**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Листов 46

Москва  
2022

## **АННОТАЦИЯ**

В документе представлена базовая информация об архитектуре и эксплуатации программного обеспечения Цифровая платформа «Авангард» (далее – Система). Приведены требования к аппаратному и программному обеспечению Системы.

## СОДЕРЖАНИЕ

1. Введение .....	6
1.1. Область применения руководства пользователя.....	6
1.2. Уровень подготовки пользователя.....	6
2. Назначение и условия применения .....	7
2.1. Требования к программному обеспечению .....	7
2.2. Требования к аппаратному обеспечению .....	7
3. Основные сведения о системе .....	8
4. Архитектура системы .....	9
4.1. Структура проекта клиентского приложения.....	9
4.2. Описание компонента Host (shell) .....	10
4.3. Описание компонента Remote (mfe).....	14
4.4. Подключение нового сервиса в Host (shell).....	14
4.5. Создание динамических микрофронтенд компонентов .....	17
5. Права доступа.....	19
5.1. Подготовительные операции.....	20
5.2. Создание права доступа .....	21
5.3. Проверка прав доступа на API бэкенде.....	22
5.4. Проверка прав доступа в SPA-приложении.....	24
5.4.1. Проверка через структурные директивы .....	25
5.4.2. Проверка через гварды .....	28
5.4.3. Проверка через сервис .....	29
6. Работа с Порталом .....	30
6.1. Вход в Систему .....	30
7. Работа с разделом «Рабочие столы» Портала. ОПИСАНИЕ ОПЕРАЦИЙ	31
7.1. Рабочие столы .....	31
7.1.1. Основной рабочий стол .....	31
7.1.2. Добавление рабочего стола .....	33
7.1.3. Изменение настроек рабочего стола .....	35
7.1.4. Удаление рабочего стола .....	35
7.1.1. Навигация по рабочим столам .....	36
7.2. Виджеты рабочего стола.....	36
7.2.1. Добавление виджета.....	36

7.2.1.1. Добавление виджета вида «Конструктор» .....	38
7.2.1.2. Добавление виджета вида «Ярлык» .....	40
7.2.2. Изменение настроек виджета .....	41
7.2.3. Удаление виджета .....	42
7.2.4. Изменение размещения виджетов на рабочем столе.....	42
7.3. Настройка данных для виджетов .....	43
7.3.1. Настройка источников данных .....	43
7.3.1.1. Добавление нового источника данных .....	44
7.3.1.2. Изменение параметров источника данных.....	45
7.3.1.3. Удаление источника данных.....	45
7.3.2. Настройка запросов к источникам данных.....	45

## ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

Сокращение	Расшифровка
Система	Цифровая платформа «Авангард»
Портал	Клиентская часть Системы

## **1. ВВЕДЕНИЕ**

### **1.1. Область применения руководства пользователя**

Настоящее руководство включает основные сведения, необходимые для ознакомления с Цифровой платформой «Авангард» (далее – Система).

### **1.2. Уровень подготовки пользователя**

Настоящее Руководство пользователя ориентировано на пользователей, владеющих навыками разработки web-приложений.

## **2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ**

### **2.1. Требования к программному обеспечению**

Требуемый состав программного обеспечения пользовательской рабочей станции:

- браузер Google Chrome, Yandex Browser или Mozilla Firefox последней, или предпоследней версии.

### **2.2. Требования к аппаратному обеспечению**

К аппаратной части рабочей станции пользователя предъявляются следующие требования:

- Процессоры:
  - количество не менее 1;
  - архитектура процессора x86-64;
  - ядер не менее 2.
- Оперативная память:
  - объем не менее 4 Гб (рекомендуется 8Гб);
  - тип оперативной памяти - DDR3/DDR4/DDR5.
- Сетевой интерфейс:
  - не менее 1 порта 100 МБ/с (доступ к сервисам системы со скоростью не ниже 8 Мбит/с (для быстрой загрузки приложения рекомендуется 25 Мбит/с и выше).
- Дисковая подсистема:
  - HDD с буфером обмена не менее 64 Мб либо SSD.
- Графический режим монитора:
  - 1366x768 и выше (рекомендуется 1920x1080).
- Клавиатура, мышь.

### 3. ОСНОВНЫЕ СВЕДЕНИЯ О СИСТЕМЕ

Цифровая платформа «Авангард» представляет собой решение с применением современных веб-технологий для построения информационных систем, обеспечивающих автоматизацию комплексных бизнес-процессов.

Применяемые подходы:

- микросервисная архитектура;
- динамическая компоновка интерфейса из подключаемых сервисов;
- подключение сторонних информационных систем в качестве сервисов, сохраняя при этом общие принципы функционирования платформы;
- единая точка входа (система аутентификации на базе протокола OAuth 2.0);
- единая среда управления правами доступа пользователей к функционалу подключенных сервисов;
- типовой графический интерфейс и общие способы взаимодействия с ним (UI и UX) в рамках всех подключаемых сервисов;
- повторное использование кодовой базы за счёт библиотек фронтенд компонентов (npm-пакетов) и подключаемых бэкенд модулей (git-сабмодулей).

Преимущества:

- повышение эффективности труда работников благодаря использованию единого информационного пространства с типовыми интерфейсными решениями;
- обеспечение возможности доработки функционала поставляемых готовых сервисов под специфические требования Заказчика;
- поэтапная модернизация информационного пространства организации без необходимости срочного вывода из эксплуатации устаревших информационных систем;
- ускоренная разработка новых сервисов за счёт повторного использования кодовой базы.



## 4. АРХИТЕКТУРА СИСТЕМЫ

В концепцию Системы положена микросервисная архитектура с применением микрофронтендов.

Для сборки единого приложения из разных микросервисов используется плагин для Webpack 5 Module Federation.

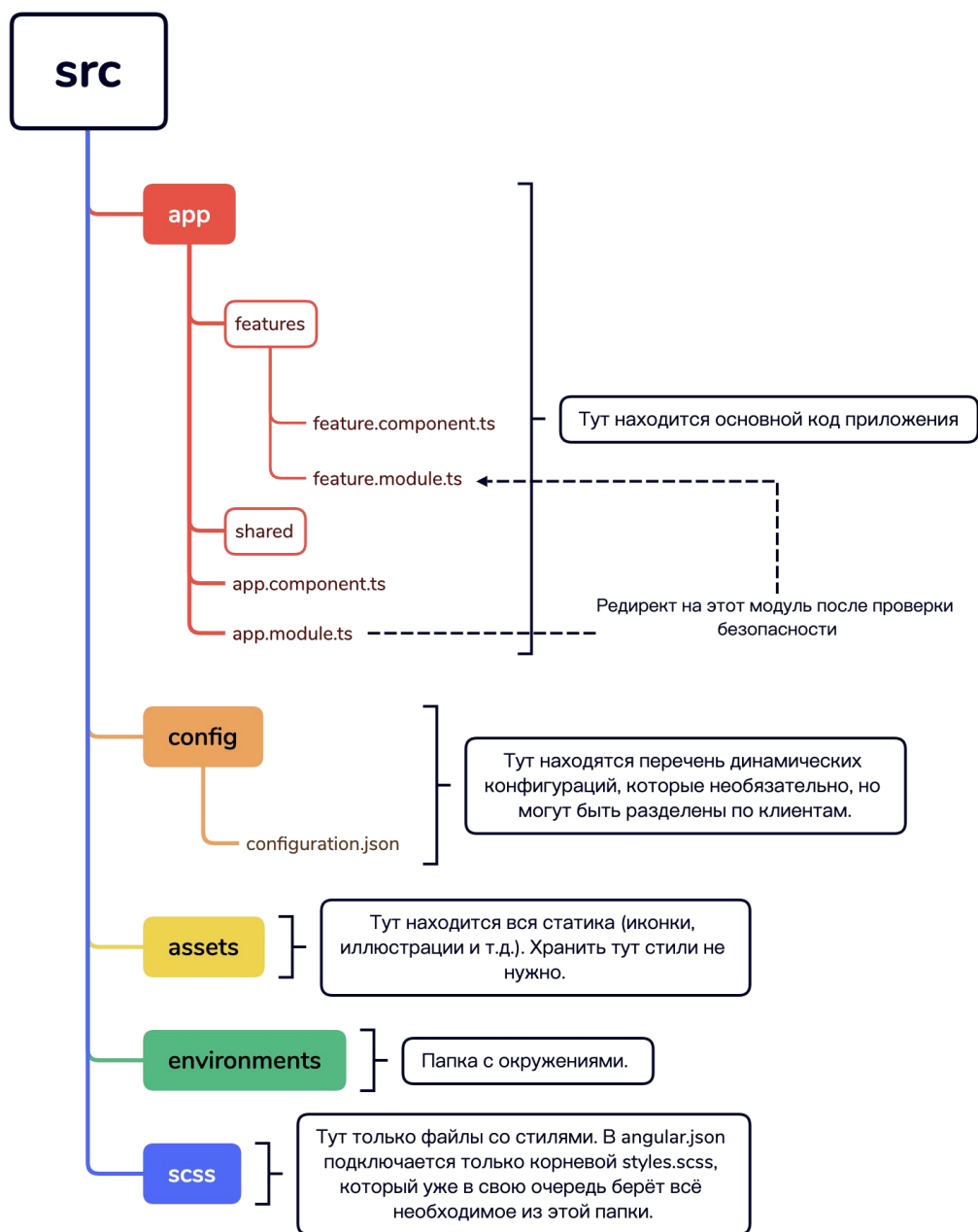
Архитектура микрофронтенда включает компоненты Host и Remote. Host обеспечивает загрузку и отображение контента других микросервисов (далее – shell). Remote обеспечивает передачу контента другому микросервису (далее – mfe). Концептуальная схема архитектуры:



Рис. 4.1

### 4.1. Структура проекта клиентского приложения

Структура проекта клиентского приложения для микрофронтенда приведена на рисунке ниже (Рис. 4.2).



Presented with XMind

Рис. 4.2

## 4.2. Описание компонента Host (shell)

Shell является тонким клиентом, который содержит минимум кода, необходимого для работы общей инфраструктурой. Сюда входит интеграция с Hub (аутентификация), общий роутинг, layout (навигация), организация и управлением состоянием, а также работа с динамической конфигурацией.

Состав доступных микрофронтендов определяется через конфигурационный файл `configuration.json`, который инициализирует класс `RuntimeConfig` на этапе bootstrap приложения.

Пример содержания конфигурационного файла:

```

{
  "modules": {
    "bi": {
      "serverUrl": "http://192.168.48.124:3100",
      "path": "bi",
      "remoteEntry": "http://192.168.48.124:3100/remoteEntry.js",
      "exposedModule": "./FeaturesModule",
      "key": "FeaturesModule",
      "name": "BI"
    },
    "pm": {
      "serverUrl": "http://192.168.48.124:3110",
      "path": "pm",
      "remoteEntry": "http://192.168.48.124:3110/remoteEntry.js",
      "exposedModule": "./FeaturesModule",
      "key": "FeaturesModule",
      "name": "ProjectManagement"
    },
    "ds": {
      "serverUrl": "http://192.168.48.124:3120",
      "path": "ds",
      "remoteEntry": "http://192.168.48.124:3120/remoteEntry.js",
      "exposedModule": "./FeaturesModule",
      "key": "FeaturesModule",
      "name": "Documentation",
      "fileStorage": {
        "fileStorageUrl": "http://localhost:5120/api/file-storage-proxy/",
        "maxAllowedFileSize": 1073741824
      }
    },
    "pb": {
      "serverUrl": "http://192.168.48.124:3130",
      "path": "pb",
      "remoteEntry": "http://192.168.48.124:3130/remoteEntry.js",
      "exposedModule": "./FeaturesModule",
      "key": "FeaturesModule",
      "name": "Phonebook"
    }
  }
}

```

```

"notes": {
  "serverUrl": "http://192.168.48.124:3140",
  "path": "notes",
  "remoteEntry": "http://192.168.48.124:3140/remoteEntry.js",
  "exposedModule": "./FeaturesModule",
  "key": "FeaturesModule",
  "name": "Notes"
},
"timesheet": {
  "serverUrl": "http://192.168.48.124:3150",
  "path": "timesheet",
  "remoteEntry": "http://192.168.48.124:3150/remoteEntry.js",
  "exposedModule": "./FeaturesModule",
  "key": "FeaturesModule",
  "name": "Timesheet"
},
"notifications": {
  "serverUrl": "http://192.168.48.124:3160",
  "path": "notifications",
  "remoteEntry": "http://192.168.48.124:3160/remoteEntry.js",
  "exposedModule": "./FeaturesModule",
  "key": "FeaturesModule",
  "name": "Notifications"
},
"messages": {
  "serverUrl": "http://192.168.48.124:3170",
  "path": "messages",
  "remoteEntry": "http://192.168.48.124:3170/remoteEntry.js",
  "exposedModule": "./FeaturesModule",
  "key": "FeaturesModule",
  "name": "Messages"
}
},
"authentication": {
  "authority": "https://192.168.48.124:1090",
  "redirect_uri": "http://localhost:3200/callback",
  "post_logout_redirect_uri": "http://localhost:3200/login",
  "silent_redirect_uri": "http://localhost:3200/silent-callback",
  "client_id": "ws",

```

```

    "response_type": "id_token token",
    "scope": "openid profile identityServer",
    "useLocalStorage": false,
    "automaticSilentRenew": true
  }
}

```

Динамическая конфигурация едина для оболочки и всех её составных модулей. `RuntimeConfig` предоставляется библиотекой `@quarta/core/config`. Эта библиотека включена в список переиспользуемых библиотек в `webpack.config.js`.

На основании этого конфигурационного файла динамически строится роутинг в shell:

```

@NgModule({
  imports: [
    RouterModule
  ]
})
export class ShellModule {
  constructor(
    router: Router,
    runtimeConfig: RuntimeConfig,
    loaderService: LoaderService,
    dialogService: DialogService
  ) {
    const loadStart = () => loaderService.show();
    const loadError = (m?: RemoteModule) => {
      loaderService.hide();
      dialogService.asideMessage(`Сервис ${m?.name} сейчас
недоступен. Попробуйте, пожалуйста, позже.`, true);
    };

    // Так как микрофронтенды у нас динамические, то и роутинг
    // обязательно должен быть динамическим
    // Мы должны иметь возможность добавить новый сервис без
    // сборки оболочки (только путём редактирования json)
    const routes: Routes = [
      {

```

```

        path: '',
        canActivate: [AuthenticationActivationGuard],
        children: [
            ...getRemoteRoutes(runtimeConfig, loadStart,
undefined, loadError),
            {
                path: '',
                redirectTo: 'bi',
                pathMatch: 'full'
            }
        ]
    }
];
router.resetConfig([...router.config, ...routes]);
}
}

```

### 4.3. Описание компонента Remote (mfe)

Для каждого mfe используется двухмодульный подход, который подразумевает разделение всего приложения на два комплексных модуля: `app.module.ts` (далее – **АМ**) и `features.module.ts` (далее – **ФМ**).

**АМ** содержит все вложенные модули, которые не могут быть предоставлены (single-per-application модули), а также весь код, который необходим для запуска компонента (аутентификация, навигация и т.д.). Задача роутинга в рамках данного модуля — переход к **ФМ** после проверок по безопасности.

**ФМ** содержит всё остальное. Этот модуль становится предоставляемым (типа `exposed`) и прописывается в `webpack.config.js`. Модуль является корневым для сервиса (`app.module.ts`) когда код отдаётся наружу. Роутинг определяет пути, по которым можно получить доступ непосредственно к функционалу. Если приложение должно включать singleton сервисы, то их нужно добавить в этот модуль.

### 4.4. Подключение нового сервиса в Host (shell)

Для начала нужно установить пакет `@angular-architects/module-federation`, выполнив команду:

```
npm i -D @angular-architects/module-federation
```

Далее произвести конфигурацию проекта:

```
ng add @angular-architects/module-federation --project  
YOUR_PROJECT_NAME
```

В результате будут произведены изменения в `angular.json`, появится новый файл `bootstrap.ts`, а также два конфигурационных файла `webpack.config.js` и `webpack.prod.config.js`.

Пример содержания конфигурационного файла:

```
const ModuleFederationPlugin =  
require('webpack/lib/container/ModuleFederationPlugin');  
const mf = require('@angular-architects/module-federation/webpack');  
const path = require('path');  
const share = mf.share;  
  
const sharedMappings = new mf.SharedMappings();  
sharedMappings.register(  
  path.join(__dirname, 'tsconfig.json'),  
  [/* mapped paths to share */]);  
  
module.exports = {  
  output: {  
    uniqueName: 'yourAppName',  
    publicPath: 'auto'  
  },  
  optimization: {  
    runtimeChunk: false  
  },  
  resolve: {  
    alias: {  
      ...sharedMappings.getAliases(),  
    }  
  },  
  experiments: {  
    outputModule: true  
  },  
  plugins: [  
    new ModuleFederationPlugin({  
      library: {type: 'module'},
```

```

        // For remotes (please adjust)
        // name: "yourAppName",
        // filename: "remoteEntry.js",
        // exposes: {
        //     './Component': './src/app/app.component.ts',
        // },

        // For hosts (please adjust)
        // remotes: {
        //     "mfe1": "http://localhost:3000/remoteEntry.js",
        // },

        shared: share({
            '@angular/core': {singleton: true, strictVersion:
true, requiredVersion: 'auto'},
            '@angular/common': {singleton: true, strictVersion:
true, requiredVersion: 'auto'},
            '@angular/common/http': {singleton: true,
strictVersion: true, requiredVersion: 'auto'},
            '@angular/router': {singleton: true, strictVersion:
true, requiredVersion: 'auto'},

            ...sharedMappings.getDescriptors()
        })

    )),
    sharedMappings.getPlugin()
],
};

```

Рассмотрим следующую секцию конфигурационного файла:

```

// For remotes (please adjust)
// name: "yourAppName",
// filename: "remoteEntry.js",
// exposes: {
//     './Component': './src/app/app.component.ts',
// },

```



Ключ каждого exposed модуля путь должен начинаться с символов «./», например: «./FeaturesModule», «./ListComponent».

В данной секции должны быть перечислены модули Системы, которые должны быть выделены в отдельный чанк при сборке.

Новый сервис должен быть добавлен в конфигурационный файл `configuration.json` в `shell`.

#### 4.5. Создание динамических микрофронтенд компонентов

Для обеспечения переиспользования кода используются динамические компоненты:

```
import { Component, OnInit, Type, ViewContainerRef } from
  '@angular/core';
import { loadRemoteModule } from "@angular-architects/module-
  federation";

@Component({
  selector: 'app-dynamic-example',
  template: ''
})
export class DynamicExampleComponent implements OnInit {
  constructor(
    private vcr: ViewContainerRef
  ) {}

  ngOnInit(): void {
    loadRemoteModule({
      remoteEntry: "http://localhost:3200/remoteEntry.js",
      exposedModule: "./ExposedComponent",
      type: "module"
    }).then(m => {
      const key = Object.keys(m)?.[0];
      if (key && m[key] instanceof Type) {
        const c = this.vcr.createComponent(m[key]);
        c.changeDetectorRef.detectChanges();
      }
    });
  }
}
```

Exposed модуль должен быть подготовлен для микрофронтенда.

До Angular 14 в декоратор `@Component` нужно добавлять всё необходимое в секцию `providers`.

В Angular 14 появились `standalone` компоненты, которые в этом плане идеально подходят для микрофронтенда.

## 5. ПРАВА ДОСТУПА

Проверка прав доступа пользователя основана на использовании claim-based авторизации.

Приходя в Приложение, пользователь приносит с собой Удостоверение содержащее определенный набор Claims (Утверждений) описывающих его. Гарантом действительности этих данных выступает третья сторона, которой доверяет, как Пользователь, так и Приложение.

Набор Claims состоит из пары type-value, где type описывает какой-то атрибут пользователя, а value описывает одно или несколько возможных вариантов значений.

Пример стандартного набора Claims для пользователя:

```
auth_time:1667207451
e-mail:"
avdeevas@quarta.su
"
family_name:"Test"
given_name:"Test"
idp:"local"
middle_name:"Test"
name:"Test"
sub:"d0115f55-961c-4ee8-a094-10fec46e8fa5"
"nbf": 1667236897,
"exp": 1667237197,
"iss": "
https://192.168.48.124:1090"
,
"aud": "admin",
"nonce": "0b468483553640ab88433a26a337cdc9",
"iat": 1667236897,
"s_hash":"9SmuP6Ef6Cws8fRJ5NyJqw"
"at_hash": "001qdL-3phQ_I0kBe3yF2w",
"sid": "4C899873E620DBBE25A4DE40223833E5",
"amr": ["pwd"]
```

Пример содержит не все возможные атрибуты. В зависимости от назначения, отдельные атрибуты (например, sub, iss, nbf, aud) могут регламентироваться спецификациями, например, JWT.

С точки зрения разработчика, это равноценно зарезервированному имени, которое не следует заполнять прикладными данными.

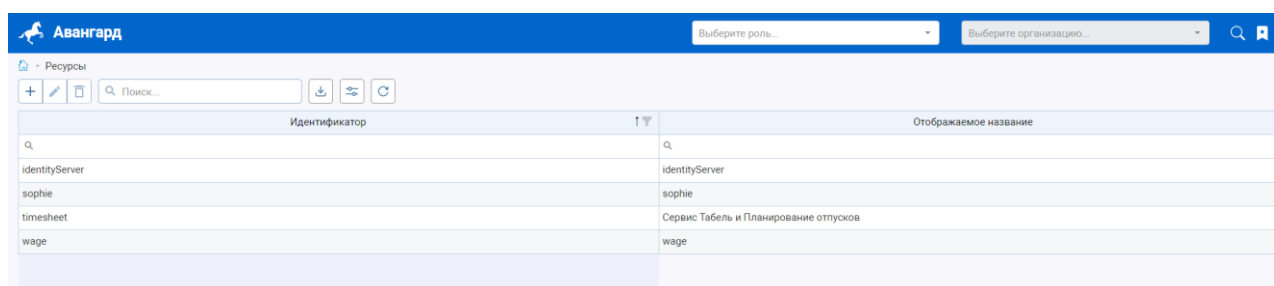
Принятым паттерном проверки существования нужного утверждения является проверка существования `type`. Это более гибкое решение чем проверка флага `true/false` как значения поля с нужным `type` т.к. принято считать, что если пользователь не обладает каким-то утверждением, то его не будет в описании `Claims`.

Микросервисы (а именно, Ресурсы) предоставляют Хабу перечень `Claims`, которые они планируют использовать для проверки доступа у пользователя при обращении к предоставляемому им API.

### 5.1. Подготовительные операции

Передача `Claims` осуществляется с использованием `back-channel http` запроса от `Ресурса` к `Хабу`. Такой запрос использует Basic Auth аутентификационную схему (симметричный ключ из пары `ApiName:ApiSecret`).

Соответственно, прежде чем создавать права доступа, необходимо удостовериться что ваш `Ресурс` существует в рамках выбранного вами `Хаба`.



Идентификатор	Отображаемое название
identityServer	identityServer
sophie	sophie
timesheet	Сервис Табель и Планирование отпусков
wage	wage

Рис. 5.1

Созданный `Ресурс` должен иметь хотя бы один код аутентификации, заполняемый на форме управления ресурсом.

Коды аутентификации не отображаются на странице по соображениям безопасности. Коды зашифрованы по алгоритму SHA256.



Управление ресурсом Сервис Табель и Планирование отпусков

Наименование \*

timesheet

Отображаемое наименование \*

Сервис Табель и Планирование отпусков

Адрес

http://localhost:5000

Коды аутентификации ①

Коды аутентификации

Сохранить Заккрыть

Рис. 5.2

В описании самого `Ресурса` необходимо добавить приведенные далее сервисы на этап бутстрапа приложения. Сделано допущение, что аутентификация пользователя уже настроена.

```
builder.Services
    .AddPermissionSyncHostedService()
    .AddClaimVerification();
```

Сервис `AddPermissionSyncHostedService` выполняет передачу прав доступа, описанных в коде Ресурса в Хаб. Сервис `AddClaimVerification` добавляет сервисы для работы атрибута `[PermissionClaim()]`, используемого в контроллерах.

## 5.2. Создание права доступа

Создаваемое право доступа должно реализовывать интерфейс `IPermissionClaim`, которое описано в проекте "Quarta.Framework.Web" модуля FrameworkCore.

```
public static class HolidayPlanningClaims
{
    public const string READ_CLAIM = "timesheet.read";

    public const string MANAGE_CLAIM = "timesheet.manage";

    public static IPermissionClaim READ = new PermissionClaim
    {
        Id = new Guid("9D7FAD39-72CC-4750-AF6E-9CD2F41A2AF9"),
        Claim = READ_CLAIM,
        Enabled = true,
        Name = "Чтение графика отпусков"
    };

    public static IPermissionClaim MANAGE = new PermissionClaim
    {
        Id = new Guid("6450A9FF-479C-4275-87FD-1826700B043A"),
        Claim = MANAGE_CLAIM,
        Enabled = true,
        Name = "Управление графиком отпусков"
    };
}
```

На данном примере создаются 2 права доступа: чтение и управление. Создаваемые права должны иметь уникальные ID и Claim в рамках вашего сервиса.

Рекомендуется именовать Claim, как комбинацию сокращенного имени вашего сервиса и названия режима. Например: `timesheet.holiday_plans`, `classifier.tags`, `pm.task_list`.

Максимально допустимая длина наименования Claim составляет 1000 символов. Рекомендуется использовать только сокращенное наименование, так как это значение передается в составе атрибутов Удостоверения о пользователе, и чем оно длиннее, тем больше данных будут передаваться между SPA-приложением, Хабом, и Ресурсом.

После создания права его необходимо зарегистрировать в DI-контейнере. Проще всего это сделать, вызвав метод расширения `AddPermission` от `IServiceCollection`. Например:

```
public static class ServiceInjections
{
    public static IServiceCollection AddServices(this
IServiceCollection collection)
    {
        collection.AddPermission(HolidayPlanningClaims.READ);
        collection.AddPermission(HolidayPlanningClaims.MANAGE);

        return collection;
    }
}
```

### 5.3. Проверка прав доступа на API бэкенде

На метод контроллера накладывается атрибут `[PermissionClaim()]`, который дает инструкцию о том, что нужно проверить указанное право доступа перед доступом к методу. Например:

```
public class HolidayPlansController : Controller
{
    private readonly IHolidayPlansService _plansService;

    public HolidayPlansController(IHolidayPlansService
plansService)
    {
```

```

        _plansService = plansService;
    }

    [HttpGet]
    [PermissionClaim(HolidayPlanningClaims.READ_CLAIM)]
    public async Task<IActionResult>
    GetHolidayDaysCountProc(DateTime dateStart, DateTime dateEnd)
    {
        var result = await
        _plansService.GetHolidayDaysCountProc(dateStart, dateEnd);

        return Ok(result);
    }
}

```

В случае, если требуется проверить наличие хотя бы одного права из списка (логическое ИЛИ), то атрибут `[PermissionClaim()]` поддерживает передачу перечня прав доступа через «точку-запятую». Например:

```

[HttpGet]
[PermissionClaim($"{HolidayPlanningClaims.READ_CLAIM});{HolidayPlannin
gClaims.MANAGE_CLAIM}")]
public async Task<IActionResult> GetHolidayDaysCountProc(DateTime
dateStart, DateTime dateEnd)
{
    var result = await
    _plansService.GetHolidayDaysCountProc(dateStart, dateEnd);

    return Ok(result);
}

```

В примере выше, доступ к методу `GetHolidayDaysCountProc` осуществляется если в Разрешении пользователя есть атрибуты `HolidayPlanningClaims.READ_CLAIM` или `HolidayPlanningClaims.MANAGE_CLAIM`.

Если требуется применить логическое И при проверке прав, то это реализуется добавлением нескольких атрибутов `[PermissionClaim()]` с разными `Claim`.

C# имеет ограничение на использование строковых литералов в атрибутах, поэтому значение поля `Claim` вынесено в строковые константы `READ_CLAIM` и `MANAGE_CLAIM` соответственно.

Также можно использовать `IIdentityInfoService` напрямую, запросить профиль пользователя и вызвать метод `HasAccess`. Например:

```
[Route("api/holiday-plans")]
public class HolidayPlansController : Controller
{
    private readonly IHolidayPlansService _plansService;
    private readonly IIdentityInfoService _infoService;

    public HolidayPlansController(IHolidayPlansService
plansService, IIdentityInfoService infoService)
    {
        _plansService = plansService;
        _infoService = infoService;
    }

    [HttpGet]
    public async Task<IActionResult>
GetHolidayDaysCountProc(DateTime dateStart, DateTime dateEnd)
    {
        bool hasClaim =
_infoService.GetUserProfile().HasAccess(HolidayPlanningClaims.READ_CLA
IM);

        if (!hasClaim)
            return Unauthorized();

        var result = await
_plansService.GetHolidayDaysCountProc(dateStart, dateEnd);

        return Ok(result);
    }
}
```

#### 5.4. Проверка прав доступа в SPA-приложении

Предварительные шаги:

- `npm install quarta-authentication;`
- добавить `AuthenticationModule.forRoot(Options)` в корень



приложения;

- добавить `LoginModule.forRoot(Options)` в корень приложения.
- Проверять права доступа можно тремя способами:
- структурные директивы;
  - гварды;
  - инъекция сервиса.

#### 5.4.1. Проверка через структурные директивы

Импортируйте модуль `ClaimAccessModule` в ваш прикладной модуль. Данный модуль объявляет 2 структурные директивы, которые могут отображать или скрывать контент на странице:

1. `[quRequireClaim]` скрывает или отображает содержимый контент в зависимости от наличия Разрешения (`claim`);
2. `[quIsReadOnly]` - работает в паре с реактивными формами и блокирует всю форму в зависимости от наличия Разрешения (`claim`).

Обе компоненты принимают права доступа для проверки либо из самой компоненты через параметр `@Input`, либо из поля роутинга `data/claim_edit`. Директивы попытаются найти в описании роутинга текущего пути (включая все родительские пути, в зависимости от конфигурации приложения) поле `claim_edit` и получить из него один или набор Разрешений (`Claims`), если права не переданы из компоненты.

Использование начинается с импорта `ClaimAccessModule`:

```
import { ClaimAccessModule } from "quarta-authentication";

@NgModule({
  imports: [
    ClaimAccessModule,
  ],
  declarations: [MyComponent],
})
export class MyModule {
}
```

Пример передачи Разрешения (`claim`) из компоненты для `[quRequireClaim]`:

```
@Component()
export class MyComponent {
```

```

    public editClaim: string = 'timesheet.manage';
}

<div *quRequireClaim="editClaim">
    <button (click)="window.alert('add-clicked')" title="Добавить">
        <span class="btn-grid-panel-add"></span>
    </button>
</div>

```

Вместо одного Разрешения может передаваться массив Разрешений (**Claims**). В этом случае проверка Разрешений будет выполняться с использованием логического оператора ИЛИ.

Пример передачи Разрешения (**claim**) из роутига для **[quRequireClaim]**:

```

import { ClaimAccessModule } from "quarta-authentication";

@NgModule({
  imports: [
    ClaimAccessModule,
    RouterModule.forChild([
      {
        component: MyComponent,
        data: {
          claim_edit: 'timesheet.manage'
        },
      },
    ]),
  ],
  declarations: [MyComponent]
})
export class MyModule {
}

<div *quRequireClaim>
    <button (click)="window.alert('add-clicked')" title="Добавить">
        <span class="btn-grid-panel-add"></span>
    </button>

```

```
</div>
```

Ниже приведен пример передачи Разрешения (`claim`) из роутинга для `[quIsReadOnly]`. Директиву нужно привязывать на тот же тег что и `[formGroup]` или же передать ей в качестве `@Input()`-параметра значение `[form]`.

```
<form [formGroup]="form" qu-is-read-only>

  <!-- Some controls !-->

  <hr/>

  <div class="d-flex justify-content-start">
    <button class="btn btn-primary" (click)="save()">
      Сохранить
    </button>
    <button class="btn btn-link" (click)="onCancelClicked()">
      Закрыть
    </button>
  </div>
</form>
```

`[QuIsReadOnly]` также умеет скрывать и блокировать кнопки, находящиеся в составе формы, если они имеют класс `btn`, `btn-grid` и не имеют класс `btn-link`. На примере выше кнопка «Сохранить» будет скрыта, а кнопка «Закрыть» будет доступна.

Пример передачи Разрешения (`claim`) из компоненты для `[quIsReadOnly]`:

```
@Component()
export class MyComponent {
  public editClaim: string = 'timesheet.manage';
}

<div [quIsReadOnly]="editClaim" [form]="form">
  <form [formGroup]="form">
    <!-- Some controls !-->
    <hr/>
```

```

    <div class="d-flex justify-content-start">
      <button class="btn btn-primary" (click)="save()">
        Сохранить
      </button>
      <button class="btn btn-link" (click)="onCancelClicked()">
        Закрыть
      </button>
    </div>
  </form>
</div>

```

#### 5.4.2. Проверка через гварды

Импортируйте модуль `ClaimAccessModule` в ваш прикладной модуль. Модуль объявляет гвард `CanActivateByClaimGuard` который запускается при переходе к роуту, в котором он объявлен. Гвард ищет в дереве роутинга поле `data/claim` и если у пользователя нет ни одного из указанных значений `Claims` блокирует переход по пути.

```

import { ClaimAccessModule } from "quarta-authentication";

@NgModule({
  imports: [
    ClaimAccessModule,
    RouterModule.forChild([
      {
        component: MyComponent,
        canActivate: [CanActivateByClaimGuard],
        data: {
          claim: [
            claims.personalCardWorkActivityRead,
            claims.personalCardWorkActivityRecentExperience,
            claims.personalCardWorkActivityCurrentExperience
          ],
        },
        children: ...
      }
    ])
  ]
})

```

```

    ],
    declarations: [MyComponent],
  })
  export class MyModule {
  }

```

Объявленный гвард запускается всегда, когда происходит попытка перейти по роуту, в котором он объявлен. Если объявлена цепочка гвардов, они исполняются в порядке от вышестоящей секции роутинга к нижестоящей.

### 5.4.3. Проверка через сервис

Проинжектируйте сервис `PermissionService` (он предоставляется корневым `AuthenticationModule`) и передавайте ему `Claims` наличие которых вы хотите проверить. Если у пользователя они есть, то метод `hasAnyClaim` вернет `true`. Если нет - `false`.

```

@Component()
export class MyComponent implements OnInit {

  constructor(
    private permissionService: PermissionService
  ) {
  }

  ngOnInit(): void {

    this.permissionService.hasAnyClaim('timesheet.manage').toPromise()
      .then((doIHaveAClaim: boolean) => {

        });

  }
}

```

## 6. РАБОТА С ПОРТАЛОМ

Клиентская часть Системы (далее – Портал) обеспечивает работу пользователей с подключенными микросервисами в едином информационном пространстве через браузер. Действия по развёртыванию Портала описаны в Руководстве администратора.

По умолчанию в дистрибутив Системы входят микросервис «Рабочие столы», позволяющий создавать динамические витрины данных. Описание работы с соответствующим разделом Портала приведено в п. 7 настоящего руководства.

### 6.1. Вход в Систему

Для входа в систему необходимо:

- запустить браузер;
- в адресную строку браузера ввести адрес сервера Системы;
- в форме авторизации ввести логин и пароль, выданные администратором системы (по умолчанию ввести логин – test, пароль – 1), нажать кнопку «Войти».

Рис. 6.1

Если логин или пароль были введены неверно, то при нажатии на кнопку «Войти» на форме отобразится соответствующее уведомление.

Возможными причинами ввода некорректного значения могут быть: иной язык ввода, регистр клавиатуры, количество пробелов между символами.

Для восстановления пароля или перейдите по гиперссылке «Забыли пароль?» под полем ввода пароля и следуйте инструкциям.

Для создания учётной записи перейдите по гиперссылке «Создайте новый!» под кнопкой «Войти» и следуйте инструкциям.

## 7. РАБОТА С РАЗДЕЛОМ «РАБОЧИЕ СТОЛЫ» ПОРТАЛА. ОПИСАНИЕ ОПЕРАЦИЙ

### 7.1. Рабочие столы

#### 7.1.1. Основной рабочий стол

После входа в Систему автоматически открывается основной рабочий стол, который выступает в роли главной страницы рабочего кабинета:

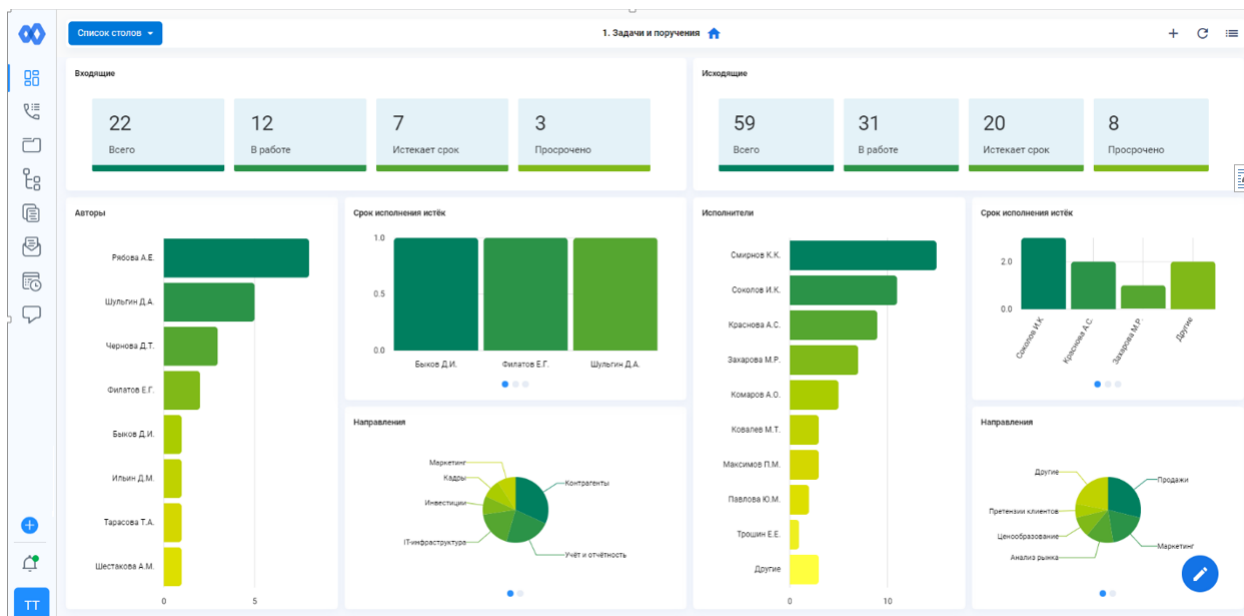


Рис. 7.1

Сверху по центру страницы отображается наименование рабочего стола. Рядом с наименованием **основного рабочего стола** отображается пиктограмма 🏠.

Слева кнопка выбора Рабочего стола 🗑️ (п.7.1.2).

Справа пиктограммы:

+ – добавление виджетов (п.7.2.1);

🔄 – обновить рабочий стол;

☰ – действия (п. 7.1.3);

Рабочий стол состоит из набора виджетов (более подробно п.7.2).

В правом нижнем углу расположена кнопка – 🖋️ редактирование рабочего стола

В левом нижнем углу расположена кнопка с настройками:

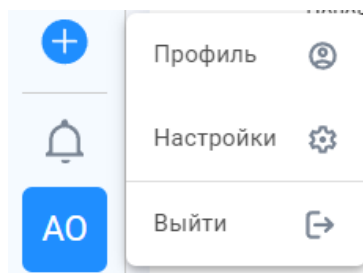


Рис. 7.2

Если основной рабочий стол не задан, то отобразится страница:




### Основной рабочий стол не задан

Основной рабочий стол — это точка входа, на которую будет осуществляться редирект при входе  
Любой рабочий стол может быть определён в качестве основного  
Вы можете переопределить это в любой момент времени

[Перейти к списку рабочих столов](#)

Рис. 7.3

Чтобы рабочий стол сделать основным, нужно его открыть (см. п. 7.1.1), кликнуть сверху справа на пиктограмму  и выбрать пункт «Сделать основным»:

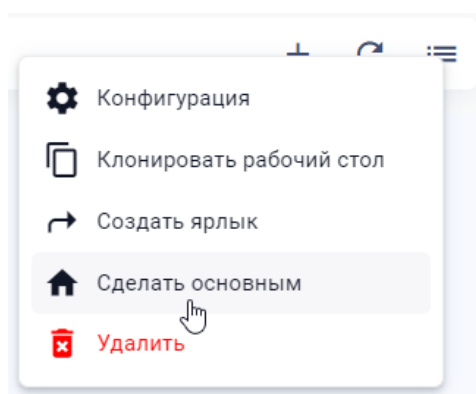




Рис. 7.4

На панели около наименования рабочего стола появится пиктограмма , свидетельствующая, что выбранный стол является основным.



### 7.1.2. Добавление рабочего стола

Добавить рабочий стол можно несколькими способами:

- на странице рабочего стола нажать клавишу F1;
- в списке рабочих столов перейти по гиперссылке «Добавить новый рабочий стол» (см. Рис. 7.12);
- на странице рабочего стола кликнуть сверху справа на пиктограмму  и выбрать пункт «Клонировать рабочий стол»:

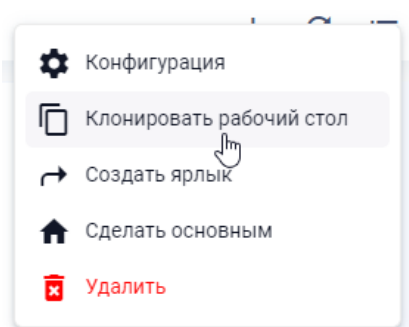


Рис. 7.5

При выборе одного из первых двух способов открывается форма добавления нового рабочего стола:

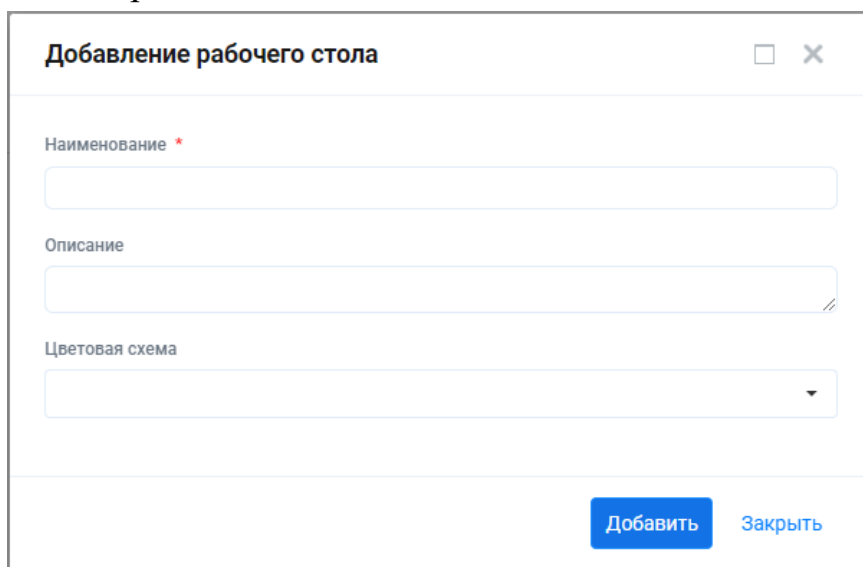
Изображение показывает форму «Добавление рабочего стола». В форме три поля для ввода: «Наименование \*» (обязательное), «Описание» и «Цветовая схема». Внизу формы находятся две кнопки: «Добавить» (синяя) и «Закрыть» (серая).

Рис. 7.6

После заполнения полей и нажатия на кнопку «Добавить» отобразится уведомление о выполнении операции и новый рабочий стол добавится в список рабочих столов (см. Рис. 7.12). Добавленный рабочий стол будет пустым (без виджетов):

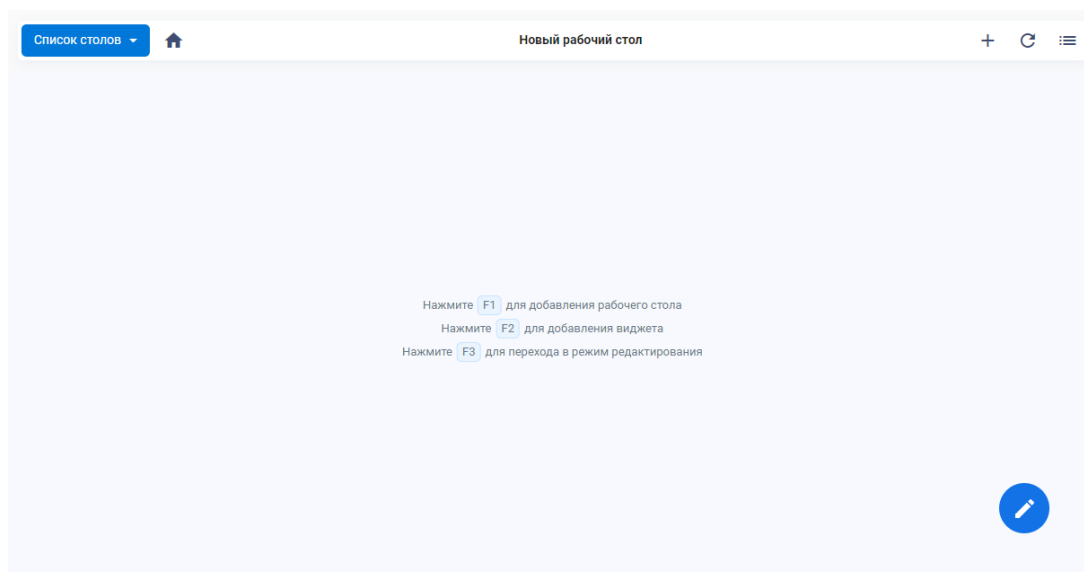


Рис. 7.7


Процедура добавления нового виджета рабочего стола описана в п. 7.2.1.

Клонирование рабочего стола подразумевает создание нового рабочего стола с копированием всех виджетов из активного рабочего стола. При выборе этого варианта открывается форма клонирования рабочего стола:

Рис. 7.8

Внести требуемые изменения и нажать кнопку «Клонировать». Отобразится уведомление о выполнении операции и в список рабочих столов будет добавлена новая запись с указанным наименованием.

### 7.1.3. Изменение настроек рабочего стола

Чтобы изменить настройки рабочего стола, нужно его открыть (см. п. 7.1.1), кликнуть сверху справа на пиктограмму  и выбрать пункт «Конфигурация»:

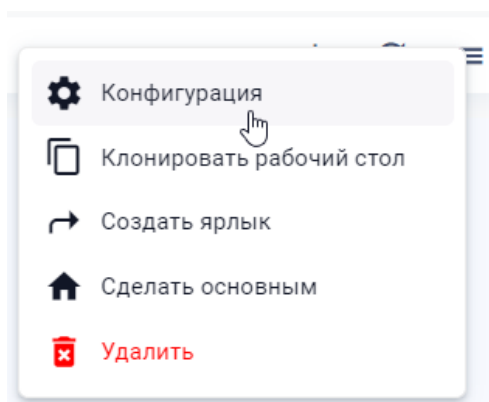


Рис. 7.9

Откроется форма редактирования настроек рабочего стола:

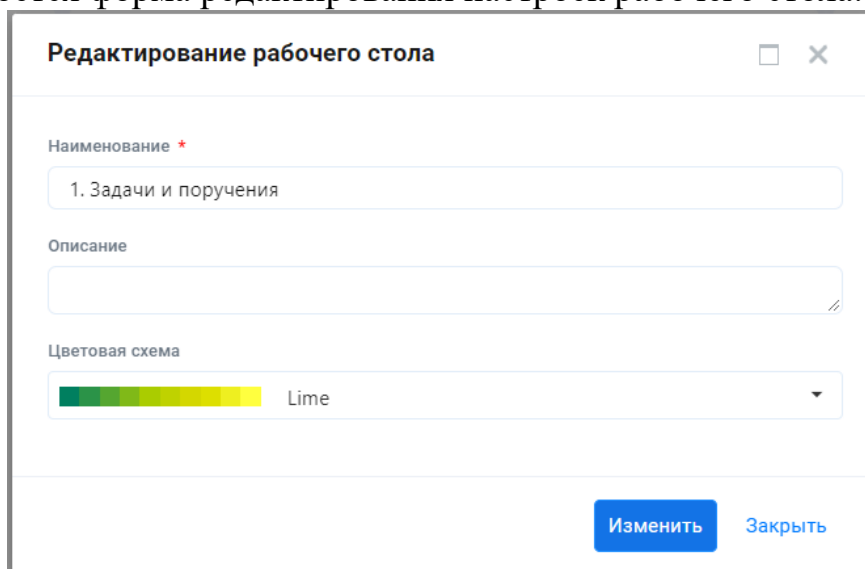



Рис. 7.10

Внести требуемые изменения и нажать кнопку «Изменить».

### 7.1.4. Удаление рабочего стола

Чтобы удалить рабочий стол, нужно его открыть (см. п. 7.1.1), кликнуть сверху справа на пиктограмму  и выбрать пункт «Удалить» (см. Рис. 7.9). После чего подтвердить удаление в открывшемся диалоговом окне:

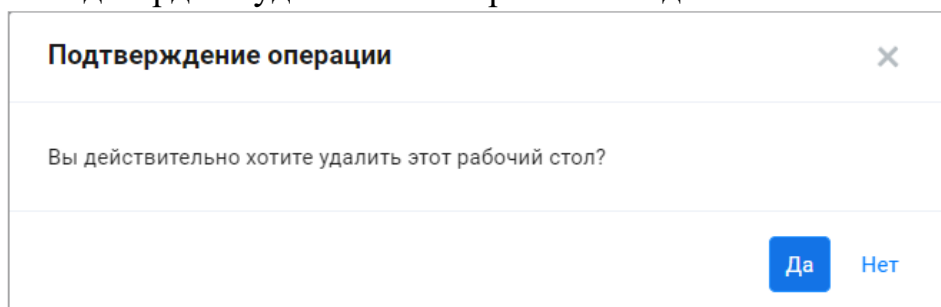


Рис. 7.11

### 7.1.1. Навигация по рабочим столам

Список доступных рабочих столов отображается при нажатии на кнопку «Список столов»:

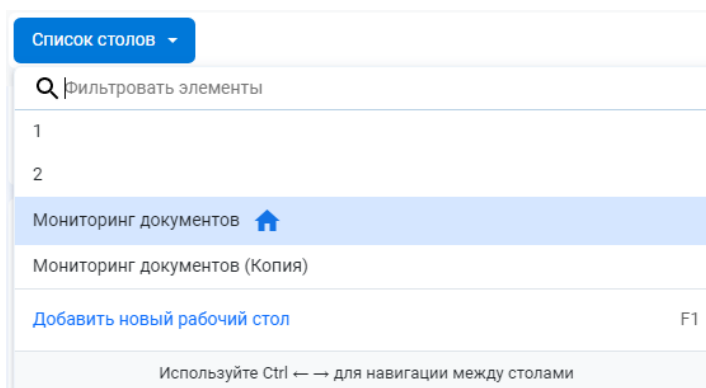


Рис. 7.12

Для перехода к другому рабочему столу его необходимо выбрать в выпадающем списке кликом по левой кнопке мыши. Для перехода к следующему или предыдущему рабочему столу по списку можно воспользоваться сочетанием клавиш «Ctrl + →» или «Ctrl + ←» соответственно.


Список рабочих столов является динамическим. Можно добавлять и удалять рабочие столы (см. п. 7.1.2).

## 7.2. Виджеты рабочего стола

Виджеты рабочего стола обеспечивают оперативный доступ к требуемой информации для целевой группы пользователей рабочего стола. Также с помощью виджетов можно сформировать набор гиперссылок для обеспечения быстрого перехода на внутренние и внешние веб-ресурсы.

### 7.2.1. Добавление виджета

Добавить виджет рабочий стол можно несколькими способами:

- на странице рабочего стола нажать клавишу F2;
- на странице рабочего стола кликнуть сверху справа на пиктограмму . На странице появится панель «Виджеты»:

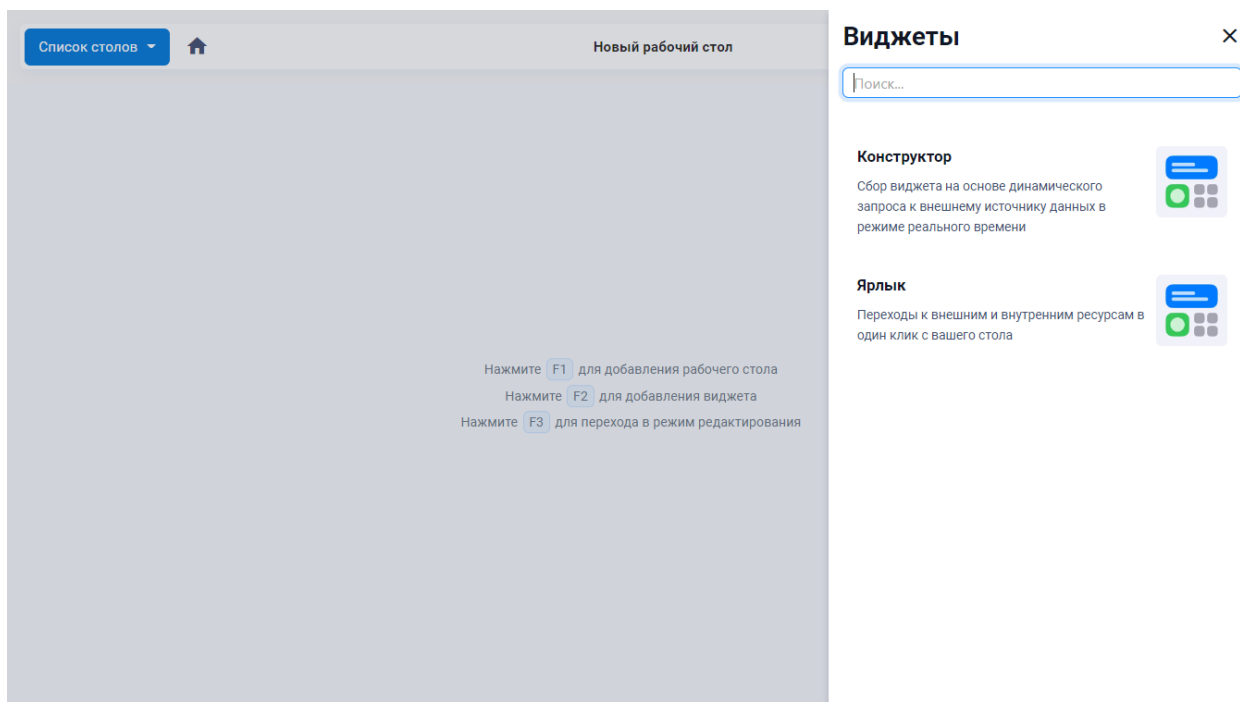


Рис. 7.13

Для добавления доступны следующие виды виджетов:

- Конструктор – отображение информации в графическом виде из предварительно настроенных источников данных на основе динамического запроса;
- Ярлык – сохранение гиперссылки с настройкой графического представления ярлыка.

Виджеты видов «Конструктор» и «Ярлык» имеют свои персональные настройки и их добавление осуществляется через специальные экранные формы. После добавления виджета его настройки можно изменить (см. п. 7.2.2).

Далее приведено описание действий по добавлению виджетов вида «Конструктор» и «Ярлык».

### 7.2.1.1. Добавление виджета вида «Конструктор»

После выбора этого вида виджетов откроется форма:

Добавление виджета

Наименование \*

Рабочий стол \*

1. Задачи и поручения

Запрос \*

Результаты

Элементов на страницу: 10 0 из 0 |< < > >|

Добавить Закрыть

Рис. 7.14

Заполните наименование нового виджета и выберите из списка требуемые запрос, результат выполнения которого сформирует набор данных для отображения в виджете.

**Примечание.** Перечень запросов к источникам данных должен быть сформирован заранее (см. п. 7.3).

После выбора запроса он автоматически исполнится и его результаты в табличном виде отобразятся на вкладке «Результаты»:

**Добавление виджета**

Наименование \*  
Распределение по группам стажей

Рабочий стол \*  
Новый рабочий стол

Запрос \*  
Количество - По стажу

Преднастроенные визуализации  
Select...

**Результаты**

Стаж System.String	Количество System.Int64
1. Менее года	16
2. От 1 года до 5 лет	27
3. От 5 до 10 лет	20
4. От 10 до 15 лет	18
5. Более 15 лет	26

Добавить    Закрыть

Рис. 7.15

На данном этапе можно проверить корректность выбранного запроса и результатов его выполнения.

Далее необходимо выбрать доступное для данного запроса визуальное представление данных в поле «Преднастроенные визуализации» и нажать кнопку «Добавить».

**Примечание.** Можно выбрать сразу несколько визуальных представлений данных. В этом случае в виджете на рабочем столе появится возможность переключения визуального представления данных.

После нажатия на кнопку «Добавить», если были заполнены все обязательные поля, экранная форма автоматически закроется и на рабочем столе появится добавленный виджет:

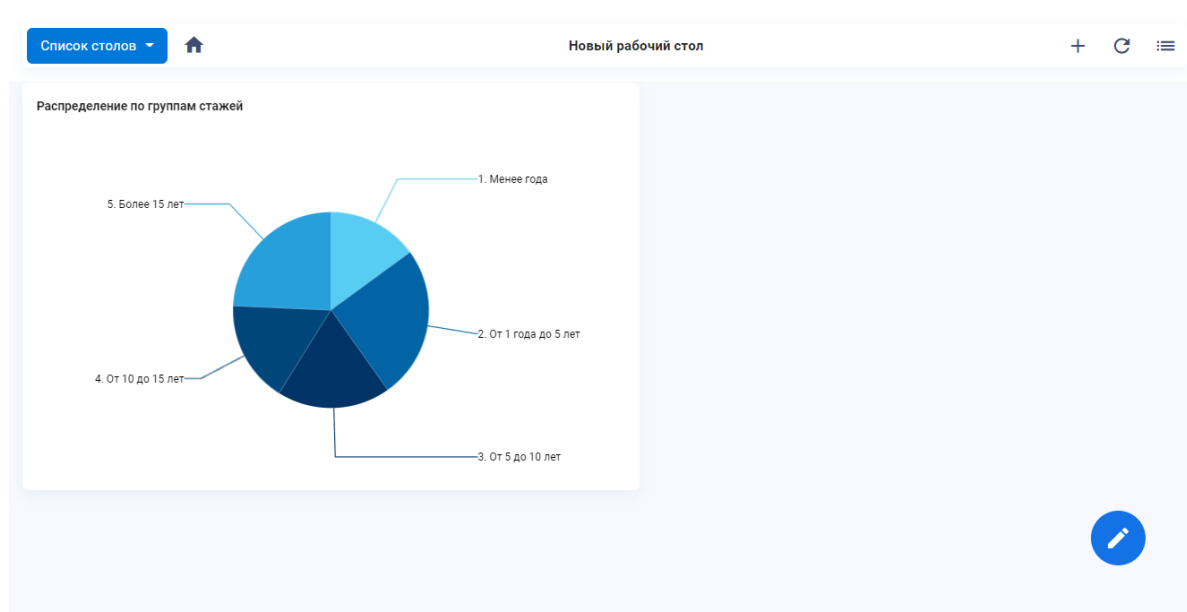


Рис. 7.16

#### 7.2.1.2. Добавление виджета вида «Ярлык»

После выбора этого вида виджетов откроется форма:

Рис. 7.17

Окно формы содержит следующие поля:

- **«Наименование»** — наименование виджета для отображения во всплывающей подсказке;

**Примечание.** Если не будет задана картинка заднего фона, то первые буквы начальных слов наименования будут использованы для отображения в видимой части виджета на рабочем столе.



- **«Цвет фона»** – задание цвета фона виджета путём выбора из цветовой палитры или ввода кода цвета.
- **«Цвет текста»** – задание текста наименования виджета путём выбора из цветовой палитры или ввода кода цвета.
- **«Ссылка»** – задание гиперссылки на внутренний или внешний веб-ресурс.
- **«Задний фон»** – задание фоновой картинки для отображения в видимой части виджета на рабочем столе.

В правой части формы отображается визуальное представление виджета в соответствии с заданными настройками. В таком виде виджет будет добавлен на рабочий стол. Пример формы с заполненными полями:

Добавление виджета

Наименование \*

Центральный Банк России Новости

Цвет фона

Цвет текста

Ссылка \*

<https://www.cbr.ru/news/>

Задний фон

<https://www.cbr.ru/common/images/logostyle/log>

Применить Закрыть

Рис. 7.18

### 7.2.2. Изменение настроек виджета

Не все виды виджетов имеют свои персональные настройки. Если настройки добавленного виджета можно изменить, то в списке доступных действий для него отображается пункт «Конфигурация»:

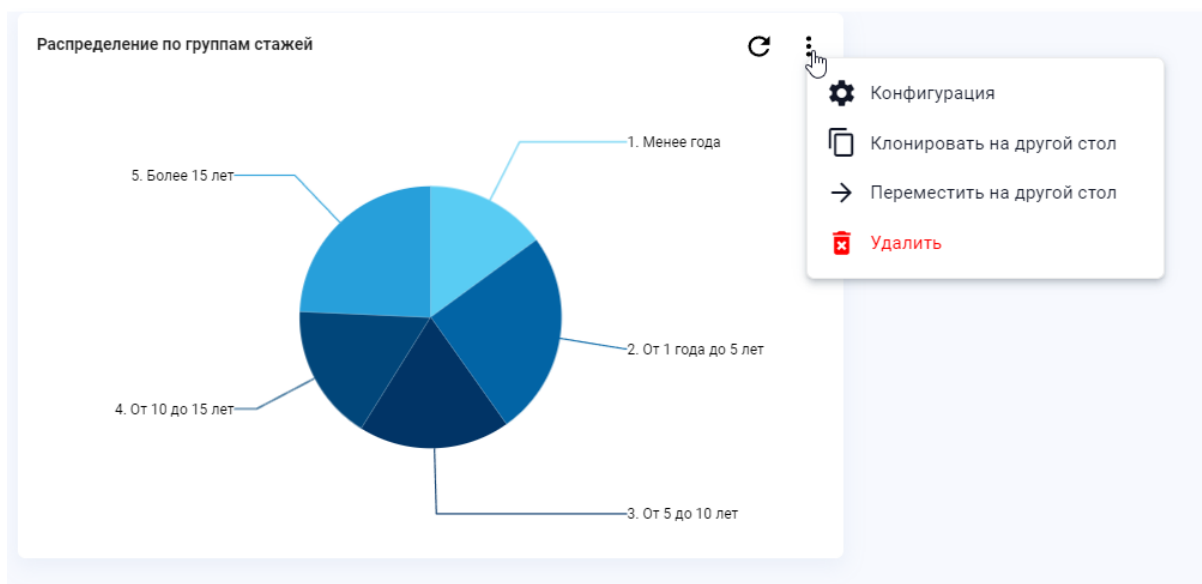




Рис. 7.19

После выбора пункта «Конфигурация» откроется форма изменения настроек виджета, соответствующая его виду. Внесите требуемые изменения и нажмите кнопку «Применить». Состояние виджета на рабочем столе автоматически обновится, применятся внесённые изменения.

### 7.2.3. Удаление виджета

Для удаления виджета с рабочего стола необходимо в его списке доступных действий выбрать пункт «Удалить» (см. п. Рис. 7.19).

### 7.2.4. Изменение размещения виджетов на рабочем столе

Для изменения  змещения виджетов на рабочем столе необходимо кликнуть по кнопке  снизу справа рабочего стола. Рабочий стол перейдёт в режим макетирования:

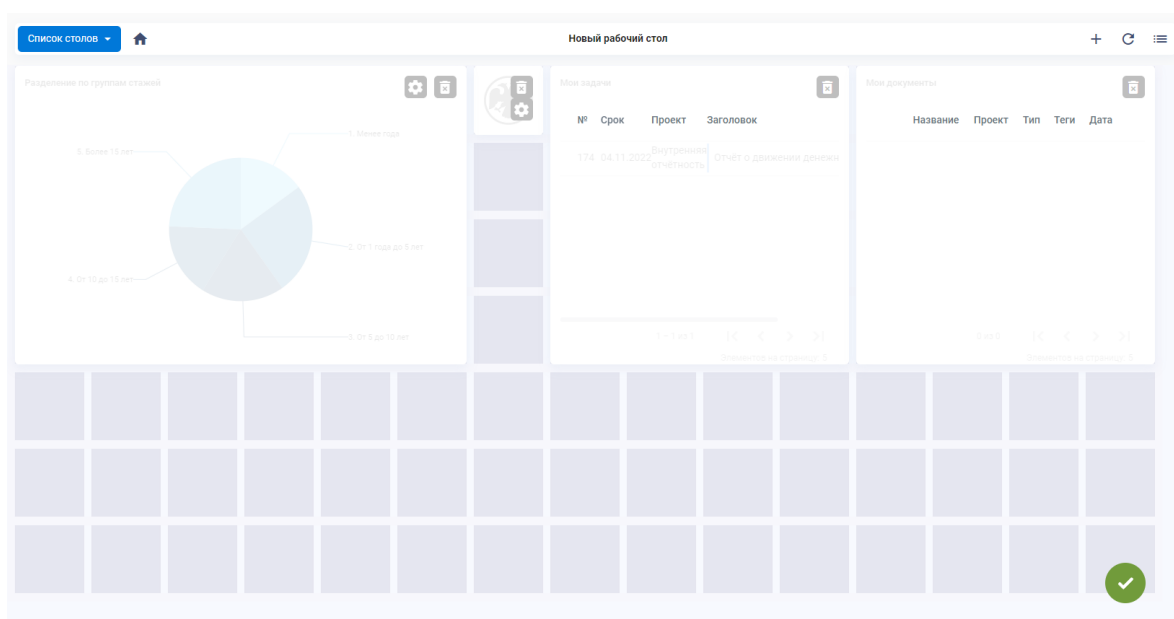





Рис. 7.20

В режиме макетирования доступны следующие возможности:

- перемещение виджетов по размерной сетке курсором мыши, при этом курсор принимает следующий вид: ;
- изменение размеров виджетов (за исключением виджетов вида «Ярлык») по узлам размерной сетки курсором мыши, при этом курсор принимает следующий вид:  (угол наклона курсора может отличаться от приведённого).

Также в этом режиме для виджетов отображаются пиктограммы изменения настроек и удаления.

Внести требуемые изменения и кликнуть по пиктограмме для применения изменений и возврата в режим просмотра рабочего стола. 

### 7.3. Настройка данных для виджетов

Настройка данных для виджетов заключается в определении источников данных и формирования запросов данных к источникам.



#### 7.3.1. Настройка источников данных


Для формирования перечня источников данных необходимо в браузере открыть страницу «<https://hostname/app/bi/data-source>»:

Список источников данных

Определите источники данных для запросов.

Новый источник



 Поиск...

<input type="checkbox"/>	Наименование ↑	Тип	Хост	Порт	База данных
<input type="checkbox"/>	1С:Розница 8		192.168.10.11	1433	retail_storage
<input type="checkbox"/>	Корпоративный портал		192.168.10.12	5433	orgsite_storage
<input type="checkbox"/>	Кадры		192.168.10.13	1433	hr_storage
<input type="checkbox"/>	Бухгалтерия		192.168.10.14	5433	accounting_storage

Элементов на страницу: 20

1 – 4 из 4







Рис. 7.21

**Примечание.** По умолчанию список источников может быть пустым. Картинка выше приведена только в качестве примера.

### 7.3.1.1. Добавление нового источника данных

Нажмите кнопку «Новый источник» откроется форма:

Добавление источника данных

Поставщик данных \*

Наименование \*

Хост \*

Порт \*

Пользователь \*

Пароль \*

База данных \*

Проверить соединение Добавить Заккрыть

Рис. 7.22

Окно формы содержит следующие поля:

- **«Поставщик данных»** – выберите из выпадающего списка СУБД сервера базы данных;
- **«Наименование»** – задание наименования источника данных для списка источников данных;
- **«Хост»** – задание имя сервера;
- **«Порт»** – задание номера порта на сервере для подключения;
- **«Пользователь»** – задание имени пользователя для подключения к выбранному поставщику данных;
- **«Пароль»** – задание пароля пользователя для подключения к выбранной СУБД;
- **«База данных»** – задание имени пользователя для подключения к выбранной СУБД.




Заполните все поля и нажмите кнопку «Проверить соединение». Отобразится уведомление о результатах установки соединения.


Нажмите кнопку «Добавить», в список источников данных будет добавлена новая запись.

### 7.3.1.2. Изменение параметров источника данных

В списке источников данных выберете запись, в параметры которой требуется внести изменения:


**Список источников данных**  
Определите источники данных для запросов.


	Наименование ↑	Тип	Хост	Порт	База данных
<input checked="" type="checkbox"/>	1С:Розница 8		192.168.10.11	1433	retail_storage
<input type="checkbox"/>	Корпоративный портал		192.168.10.12	5433	orgsite_storage
<input type="checkbox"/>	Кадры		192.168.10.13	1433	hr_storage
<input type="checkbox"/>	Бухгалтерия		192.168.10.14	5433	accounting_storage

Элементов на страницу: 20 1 – 4 из 4

Рис. 7.23

После чего нажмите кнопку . Откроется форма изменения источника данных, набор полей которой идентичен набору полей формы добавления источника данных (см. Рис. 7.22). Внесите требуемые изменения и нажмите кнопку «Сохранить».

### 7.3.1.3. Удаление источника данных

В списке источников данных выберете одну или несколько записей и нажмите кнопку . После подтверждения операции выбранные записи будут удалены.

### 7.3.2. Настройка запросов к источникам данных

Для формирования перечня запросов к источникам данных необходимо в браузере открыть страницу «<https://hostname/app/bi/query>»:

Список источников данных

Определите источники данных для запросов.

✎

🗑

↻

Поиск...

Новый источник

<input type="checkbox"/> Наименование ↑	Тип	Хост	Порт	База данных
<input type="checkbox"/> 1С:Розница 8		192.168.10.11	1433	retail_storage
<input type="checkbox"/> Корпоративный портал		192.168.10.12	5433	orgsite_storage
<input type="checkbox"/> Кадры		192.168.10.13	1433	hr_storage
<input type="checkbox"/> Бухгалтерия		192.168.10.14	5433	accounting_storage

Элементов на страницу: 20 ▾

1 – 4 из 4

⏪

<

>

⏩

Рис. 7.24

**Примечание.** По умолчанию список источников может быть пустым. Картинка выше приведена только в качестве примера.