1. ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В этом разделе подробно рассматривается функционирование программных модулей. Перечислены соответствующие классы, а также дано описание большинства их компонентов. Состав основных классов показан также на диаграмме классов (чертеж ГУИР.400201.047 РР.1).

* 1. Классы веб-приложения
     1. Класс User

Этот класс является отображением структуры данных пользователя и предоставляет методы для работы с ними.

В классе присутствуют поля и методы.

Поля:

– id – поле типа BigInt, представляющее собой целое число, содержащее в себе идентификационный номер пользователя;

– email – поле типа String, представляющее собой строку, хранящую в себе адрес электронной почты пользователя;

– encrypted\_password – поле типа String, представляющее собой строку, хранящую в себе зашифрованный пароль пользователя для авторизации в приложении;

– reset\_password\_token – поле типа String, представляющее собой строку, хранящую в себе ключ, необходимый для восстановления пароля пользователя для авторизации в приложении;

– reset\_password\_sent\_at – поле типа DateTime, представляющее собой дату и время отсылки пользователю на адрес электронной почты сообщения для восстановления пароля;

– remember\_created\_at – поле типа DateTime, представляющее собой дату и время последней установки пользователем опции «Запомнить меня» при авторизации;

– sign\_in\_count – поле типа Integer, представляет собой целое число, которое отображает количество входов пользователя в приложение;

– current\_sign\_in\_at – поле типа DateTime, представляющее собой дату и время авторизации пользователя в текущей сессии;

– last\_sign\_in\_at – поле типа DateTime, представляющее собой дату и время авторизации пользователя в последней сессии;

– current\_sign\_in\_ip – поле типа String, представляющее собой строку, хранящую в себе ip-адрес пользователя в текущей сессии;

– last\_sign\_in\_ip – поле типа String, представляющее собой строку, хранящую в себе ip-адрес пользователя в последней сессии;

– created\_at – поле типа DateTime, представляющее собой дату и время создания этого пользователя;

– updated\_at – поле типа DateTime, представляющее собой дату и время последнего обновления данных этого пользователя;

– user – поле типа Boolean, представляющее собой флаг, который указывает является ли пользователь обычным пользователем в приложении;

– admin – поле типа Boolean, представляющее собой флаг, который указывает является ли пользователь администратором в приложении;

Методы:

1. static synchronized void initAdapter(final BaseApplication appContext) – метод, который инициализирует переменную applicationContext. В качестве параметра принимает переменную типа BaseApplication, указывающую на объект запущенного приложения.

2. public static Context context() – метод, возвращающий переменную applicationContext. Основное назначения получение контекста приложения из любого места.

3. public static Resources resources() – метод, инициализирует при первом обращении и возвращает переменную resources. Данный метод используется для получения ресурсов приложения.

4. public static PreferencesManager prefs() – метод, инициализирует при первом обращении и возвращает переменную preferences. Данный метод является связующим звеном блока, отвечающим за работу с файлами общих настроек приложения, и остальными блоками приложения.

5. public static IServiceApi serverApi() – метод, инициализирует при первом обращении и возвращает переменную serviceApi. Данный метод используется для доступа к API вызовам.

6. public static void reinitApi() – метод создает новую сущность объекта IServiceApi и присваивает ее переменной serviceApi, при каждом обращении к этому методу.

.

* + 1. Класс AppAdapter

Класс представляет из себя адаптер приложения. Данный класс управляет взаимодействием основных блоков приложения.

В классе присутствуют поля и методы.

Поля:

– appliactionContext – поле типа BaseApplication, содержащее в себе контекст приложения.

– resources – поле типа Resources, представляющее собой объект для доступа к ресурсам приложения;

– preferences – поле типа PreferencesManager, является объектом класса, который предоставляет доступ к файлу общих настроек приложения;

– serviceApi – поле типа IServiceApi, является объектом содержащим в себе методы для доступа к API вызовам;

Методы:

1. static synchronized void initAdapter(final BaseApplication appContext) – метод, который инициализирует переменную applicationContext. В качестве параметра принимает переменную типа BaseApplication, указывающую на объект запущенного приложения.

2. public static Context context() – метод, возвращающий переменную applicationContext. Основное назначения получение контекста приложения из любого места.

3. public static Resources resources() – метод, инициализирует при первом обращении и возвращает переменную resources. Данный метод используется для получения ресурсов приложения.

4. public static PreferencesManager prefs() – метод, инициализирует при первом обращении и возвращает переменную preferences. Данный метод является связующим звеном блока, отвечающим за работу с файлами общих настроек приложения, и остальными блоками приложения.

5. public static IServiceApi serverApi() – метод, инициализирует при первом обращении и возвращает переменную serviceApi. Данный метод используется для доступа к API вызовам.

6. public static void reinitApi() – метод создает новую сущность объекта IServiceApi и присваивает ее переменной serviceApi, при каждом обращении к этому методу.

**3.1.3** Класс BaseActivity

Класс является абстрактным, предоставляющим абстрактные методы для реализации дочерними классами Activity. Он наследуется от класса AppCompatActivity, который является базовым для всех классов Activity, которые необходимо реализовать. Класс BaseActivity содержит поля, абстрактные методы, которые будут реализовывать дочерние классы, и переопределенные методы базового класса.

Поля:

– mBinding – поле типа Binding, представляющее из себя объект привязки данных к интерфейсу, согласно архитектуре MVVM.

Методы:

1. protected abstract int getLayoutId() – абстрактный метод, который будет возвращать идентификатор файла разметки экрана.

2. protected void onCreate(Bundle savedInstanceState) – метод, который должны переопределять все наследники AppCompatActivity. Этот метод является методом жизненного цикла Activity, его вызов происходит при создании объекта Activity. Как правило, в данном методе инициализируются все UI компоненты. В качестве параметров принимает: savedInstanceState – объект типа Bundle, который представляет собой сохраненные данные, которые использует система в жизненном цикле Activity.

3. protected void initViews() – абстрактный метод, в котором производится инициализация UI компонентов. Вызов данного метода происходит в методе onCreate базового класса, таким образом дочерним классам необходимо переопределить только этот метод для того, чтобы инициализировать свои UI компоненты.

4. protected void initTools() – абстрактный метод, в котором производится инициализация каких-либо необходимых для работы компонентов.

5. protected void initBinding() – метод, который инициализирует объект mBinding.

6. public Binding getBinding() – метод, который возвращает объект mBinding.

**3.1.4** Класс BaseFragmentActivity

Класс является наследником класса BaseActivity и реализует интерфейс IFragmentListener. Реализует метод initTools, в котором инициализирует объект mFragmentManagerHelper. Данный класс является основой для Activity, в которых будет происходить работа с Fragment. Для этого в данном классе будут реализованы такие методы для работы с фрагментами, как замена, отображение, удаление фрагмента.

Поля:

– mFragmentManagerHelper – поле типа FragmentManagerHelper, являющееся объектом класса, который позволяет осуществлять транзакции с фрагментами;

– mProgressDialog – поле типа ProgressDialog, представляющее собой объект элемента интерфейса, который отображается при загрузке каких-либо данных;

Методы:

1. protected void initTools() – переопределенный метод базового класса, в котором инициализируется объект mFragmentManagerHelper.

2. protected abstract int getFragmentConatinerId() – абстрактный метод, в реализации которого возвращается идентификатор контейнера для фрагментов.

3. public void replaceFragment(Class fragmentName) – метод для замены фрагмента в контейнере, вызывает внутри себя метод replaceFragment(String fragmentName, Bundle bundle, boolean addToBackStack) с параметрами fragmentName = fragmentName.getName(), bundle = null, addToBackStack = false. Это означает, что новому фрагменту не будут передаваться данные и данный фрагмент не будет помещен в стек обратного вызова.

4. public void replaceFragment(String fragmentName, Bundle bundle, boolean addToBackStack) – метод для замены фрагмента в контейнере. Принимает три переменные: fragmentName – имя фрагмента, на который необходимо заменить текущий фрагмент, bundle – данные, которые необходимо передать новому фрагменту, addToBackStack – значение, которое указывает необходимо ли добавлять фрагмент в стек обратного вызова. Внутри себя вызывает функцию объекта mFragmentManagerHelper replaceFragment(String fragmentName, Bundle bundle, boolean addToBackStack, int containerViewId, int showAnimationId, int hideAnimationId).

5. public void showProgressDialog(String message) – реализованный метод интерфейса IFragmentListener, который отображает диалог загрузки с сообщением, которое передается в качестве параметра message.

6. public void hideProgressDialog() – реализованный метод интерфейса IFragmentListener, в реализации которого скрывается диалог загрузки.

7. public void clearBackStack() – метод, который очищает стек обратного вызова.

**3.1.5** Класс SplashActivity

Класс является наследником класса AppCompatActivity. Так как данный класс является экраном отображения заставки при загрузке приложения, то он имеет особенности в реализации и поэтому он не наследуется от класса BaseActivity или BaseFragmentActivity. Главной задачей данного класса является отображение заставки при запуске приложения и перенаправление пользователя на главный экран приложения.

**3.1.6** Класс StartActivity

Класс, представляющий собой контейнер для фрагментов, а также позволяющий переключаться между экранами приложения(фрагментами).

Поля:

– START\_PROFILE – статическое константное поле типа String, представляющее из себя действие, которое передается в объекте типа Intent, и означающее, что необходимо запустить VPN сервис;

– DISCONNECT – статическое константное поле типа String, представляющее из себя действие, которое передается в объекте типа Intent, и означающее, что необходимо остановить VPN сервис;

– ACTION\_VIEW – статическое константное поле типа String, представляющее из себя действие, которое передается в объекте типа Intent, и означающее, что был совершен переход в приложение по ссылке;

– isFromDeepLink – поле типа boolean, представляющее из себя флаг, который дает понять было открыто приложение по ссылке или из лаунчера;

– mIsDisconnected – поле типа boolean, представляющее из себя флаг, который показывает состояние подключения к сети на данный момент;

– mLastReportTabIndex – поле типа int, в значение которого хранится индекс последнего открытого таба на экране статистики.

Методы:

1. public boolean isFromDeepLink() – метод возвращает значение переменной isFromDeepLink.

2. public void setFromDeepLink(boolean fromDeepLink) – метод, который устанавливает значение переменной isFromDeepLink. Переменная fromDeepLink – значение, которое будет присвоено переменной isFromDeepLink.

3. public int getLastTabReportIndex() – метод возвращает значение переменной mLastTabReportIndex.

4. protected int getFragmentContainerId() – переопределенный базовый метод, который возвращает значение идентификатора контейнера для фрагментов.

5. protected int getLayoutId() – переопределенный базовый метод, который возвращает значение идентификатора файла разметки экрана.

6. public void onNewIntent(Intent intent) – переопределенный базовый метод, в котором будут обрабатываться новые Intent, которые получает Activity. Принимает одну переменную: intent – объект типа Intent, который передается Activity при запуске ее из другого приложения или при открытии из уведомления.

5. protected void initViews() – реализованный метод базового класса BaseActivity, в котором инициализируется BottomNavigationView.

6. protected void onCreate(Bundle savedInstanceState) – метод жизненного цикла Activity. Так как данный метод вызывается при создании Activity, то в нем проверяется открыто ли приложение по ссылке. Также в данном методе регистрируется широковещательный приемник, который отслеживает изменение соединения. В качестве параметров принимает: savedInstanceState – объект типа Bundle, который представляет собой сохраненные данные, которые использует система в жизненном цикле Activity.

7. protected void onDestroy() – метод жизненного цикла Activity. Его вызов происходит при уничтожении Activity. В данном методе происходит отмена регистрации широковещательного приемника, который отслеживает изменение соединения.

8. public void changeFragment(final int itemId) – метод, который заменяет отображающийся Fragment. В качестве параметра передается константное значение типа int itemId, которое указывает на то, какой элемент был выбран в BottomNaviagationView.

9. public void setNavigationItemSelected(final int itemId) – метод, который принудительно устанавливает выбранным элемент из BottomNavigationView. В качестве параметра передается константное значение типа int itemId, которое указывает на то, какой элемент необходимо выбрать в BottomNaviagationView.

10. private void registerConnectivityReceiver() – метод, который регистрирует широковещательный приемник, отслеживающий изменения соединения.

11. private void unRegisterConnectivityReceiver() – метод, который отменяет регистрацию широковещательного приемника, отслеживающего изменения соединения.

12. public void isConnectChange(final boolean isConnect) – переопределенный метод обратного вызова, который вызывается при изменении состояния соединения. Параметр isConnect типа boolean: true – телефон подключен к сети, false –подключение к сети отсутствует.

**3.1.7** Класс BaseFragment

Класс является абстрактным и предоставляет абстрактные методы для реализации дочерними классами Fragment. Он наследуется от класса Fragment, который является базовым для всех классов Fragment, которые необходимо реализовать. Класс BaseFragment содержит поля, абстрактные методы, которые будут реализовывать дочерние классы, и переопределенные методы базового класса.

Поля:

– mBinding – поле типа Binding, представляющее из себя объект привязки данных к интерфейсу, согласно архитектуре MVVM.

– mFragmentListener – поле типа IFragmentListener, представляющее из себя объект интерфейса.

Методы:

1. protected abstract int getLayoutId() – абстрактный метод, который будет возвращать идентификатор файла разметки экрана.

2. public void onCreate(Bundle savedInstanceState) – метод, который является методом жизненного цикла Fragment, его вызов происходит при создании объекта Fragment. В качестве параметров принимает: savedInstanceState – объект типа Bundle, который представляет собой сохраненные данные, которые использует система в жизненном цикле Fragment.

3. public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) – метод жизненного цикла Fragment, в котором производится инициализация UI Fragment..

4. protected void initBinding(LayoutInflater inflater) – метод, который инициализирует объект mBinding. Параметр inflater типа LayoutInflater: объект для создания View из содержимого файла разметки.

5. protected void initBinding() – метод, который инициализирует объект mBinding.

6. protected void initData(Bundle savedInstanceState) – абстрактный метод, который инициализирует данные. Данный метод переопределяется наследниками BaseFragment. В качестве параметров принимает: savedInstanceState – объект типа Bundle, который представляет собой сохраненные данные, которые использует система в жизненном цикле Fragment.

7. protected void initTools() – абстрактный метод, в котором будет производится инициализация каких-либо необходимых для работы компонентов.

8. protected void initViews() – абстрактный метод, в котором будет производится инициализация UI компонентов. Вызов данного метода происходит в методе onCreateView базового класса, таким образом дочерним классам необходимо переопределить только этот метод для того, чтобы инициализировать свои UI компоненты.

9. public void onAttach(Activity activity) – метод жизненного цикла Fragment, который вызывается при первом присоединении Fragment к Activity. Параметр activity – объект типа Activity, который представляет собой контекст. В данном методе будет происходить вызов метода onAttachActivity.

10. public void onAttach(Context context) – метод жизненного цикла Fragment, который вызывается при первом присоединении Fragment к Activity. Отличие от предыдущего метода в том, что данный метод вызывается на версиях Android Marshmallow и выше. Параметр context – объект типа Context, который представляет собой контекст. В данном методе будет происходить вызов метода onAttachActivity.

10. protected void onAttachActivity(Activity activity) – метод, который производит инициализацию mFragmentListener. Вызов данного метода происходит в методе onAttach. Параметр activity – объект типа Activity. Данный объект присваивается mFragmentListener. Если объект activity не является объектом класса BaseFragmentActivity, то срабатывает икслючение.

11. public void replaceFragment(String fragmentName, Bundle bundle, boolean addToBackStack) – метод для замены фрагмента в контейнере. Принимает три переменные: fragmentName – имя фрагмента, на который необходимо заменить текущий фрагмент, bundle – данные, которые необходимо передать новому фрагменту, addToBackStack – значение, которое указывает необходимо ли добавлять фрагмент в стек обратного вызова. Внутри метода происходит вызов функции replaceFragment объекта mFragmentListener с параметрами fragmentName, bundle и addToBackStack.

12. public void showProgressDialog(String message) –метод, который вызывает внутри себя метод showProgressDialog объекта mFragmentListener с параметром message.

13. public void hideProgressDialog() – метод, который вызывает внутри себя метод hideProgressDialog объекта mFragmentListener.

**3.1.8** Класс MainFragment

Класс, представляющий собой отображение главного экрана приложения и реализующий логику работы с VPN.

Поля:

– START\_PROFILE – статическое константное поле типа String, представляющее из себя действие, которое передается в объекте типа Intent, и означающее, что необходимо запустить VPN сервис;

– DISCONNECT – статическое константное поле типа String, представляющее из себя действие, которое передается в объекте типа Intent, и означающее, что необходимо остановить VPN сервис;

– PROFILE\_RECONNECT – статическое константное поле типа String, представляющее из себя ключ для значения типа boolean, который означает, необходимо ли перезапускать VPN сервис;

– PROFILE\_FOREGROUND – статическое константное поле типа String, представляющее из себя ключ для значения типа boolean, которое означает запуск VPN сервиса, когда приложение было активно или нет;

– EXTRA\_VPN\_PROFILE\_ID – статическое константное поле типа String, представляющее из себя ключ для значения типа long, который хранит в себе ID VPN профиля;

– DIALOG\_TAG – статическое константное поле типа String, представляющее из себя тег для идентификации диалогового окна, которое запускается при выборе VPN профиля;

– INSTALL\_PKCS12 – статическое константное поле типа int, которое выступает в роли идентификатора действия, которое необходимо произвести. В данном случае – установить сертификат;

– PREPARE\_VPN\_SERVICE – статическое константное поле типа int, которое выступает в роли идентификатора действия, которое необходимо произвести. В данном случае – подготовить VPN сервис;

– USER\_CERT\_LOADER – статическое константное поле типа int, которое выступает в роли идентификатора для загрузчика;

– REQUEST\_DELAY – статическое константное поле типа int, которое хранит в себе значение для задержки отправки запроса;

– mProfileInfo – поле типа Bundle, представляющее из себя объект, в котором хранится информация о VPN профиле;

– mService – поле типа VpnStateService, представляющее из себя объект, хранящий сущность сервиса, отслеживающего состояние VPN;

– mPrevPKCSCert – поле типа String, в котором хранится предыдущий сертификат, если такой имеется;

– mPulseAnimation – поле типа Animation, представляющее из себя объект анимации, которая используется для отображения состояния VPN подключения;

– mPadding – поле типа int, которое хранит в себе значение отступа;

– mProfile – поле типа ParsedVpnProfile, представляющее из себя расшифрованный VPN профиль;

– mCertEntry – поле типа TrustedCertificateEntry, представляющее из себя сертификат, который запрашивается из хранилища сертификатов Android;

– mUserCertEntry – поле типа TrustedCertificateEntry, представляющее из себя сертификат текущего пользователя;

– mUserCertLoading – поле типа String, в котором хранится имя текущего сертификата;

– mReportHandler – поле типа Handler, представляющее из себя объект, который отправляет запросы на получение статистики по обнаруженным угрозам;

– mStorageChangedReceiver – поле типа BroadcastReceiver, представляющее из себя объект, отслеживающий состояние сертификатов в системе;

– mServiceConnection – поле типа ServiceConnection, представляющее из себя библиотеку имён изображений носов, ключом которой служит перечисление типа Noses;

– mUserCertificateLoaderCallbacks – поле типа LoaderManager.LoaderCallbacks<TrustedCertificateEntry>, представляющее из себя загрузчик сертификатов.

Методы:

1. protected int getLayoutId() – переопределенный базовый метод, который возвращает значение идентификатора файла разметки экрана.

2. public void onCreate(Bundle savedInstanceState) – метод, который является методом жизненного цикла Fragment, его вызов происходит при создании объекта Fragment. В качестве параметров принимает: savedInstanceState – объект типа Bundle, который представляет собой сохраненные данные, которые использует система в жизненном цикле Fragment.

3. private boolean isFromDeepLink() – метод, который возвращает значение, означающее было ли открыто приложение по ссылке.

4. private void setIsFromDeepLink(final boolean value) – метод, который устанавливает значение переменной, означающей было ли открыто приложение по ссылке. Параметр value – значение типа boolean, которое необходимо установить.

5. protected void initData(Bundle savedInstanceState) – переопределенный метод, который инициализирует данные. В качестве параметров принимает: savedInstanceState – объект типа Bundle, который представляет собой сохраненные данные, которые использует система в жизненном цикле Fragment.

6. protected void initViews() – реализованный метод базового класса BaseFragment, в котором инициализируются все UI компоненты.

7. private void setPaddingValue() – метод, в котором инициализируется значение поля mPadding в зависимости от размера экрана.

8. public void onStart() – метод жизненного цикла Fragment, его вызов происходит после метода onCreate. В данном методе происходит вызов метода post объекта mReportHandler.

9. public void onStop() – метод жизненного цикла Fragment, его вызов происходит перед методом onDestroy. В данном методе происходит вызов метода removeCallbacksAndMessages объекта mReportHandler.

10. public void updateViews() – метод, в котором происходит обновление всех UI компонентов.

11. private void addServiceStateListener() – метод, в котором регистрируется сервис mService.

12. private void requestReportData() – метод, в котором отправляется запрос на получение статистики по новым угрозам и происходит обработка результата запроса.

13. public void getUUIDFromIntent(final Uri uri) – метод, который получает уникальный идентификатор пользователя из Intent. Параметр uri – объект типа Uri, который представляет собой ссылку по которой открыли приложение.

14. private void copyPassToClipboard(final String pass) – метод, который копирует пароль в буфер обмена. Пароль передается в качестве параметра pass типа String.

15. public void onDestroy() – метод жизненного цикла Fragment, его вызов происходит при уничтожении объекта Fragment. В данном методе происходит отмена регистрации широковещательного приемника, а также отвязка сервиса mService.

16. public void onActivityResult(int requestCode, int resultCode, Intent data) – метод жизненного цикла Fragment, который вызывается при возврате к Fragment после того как из этого Fragment был открыт другой экран или другое приложение. Параметры: requestCode типа int хранит код запроса, resultCode типа int хранит код ответа и data – объект типа Intent хранит данные, который вернуло приложение.

17. public void stateChanged() – переопределенный метод сервиса VpnStateService, который вызывается изменении состояния VPN. В данном методе производится вывод ошибки, если таковая имеется, и обновление UI в зависимости от текущего состояния VPN.

18. private void importUserCertificate() – метод, который открывает диалоговое окно для импорта пользовательского сертификата из системы.

19. private void registerStorageChangedReceiver() – метод, в котором регистрируется широковещательный приемник, отслеживающий состояние сертификатов в системе.

20. private void unRegisterStorageChangedReceiver() – метод, в котором происходит отмена регистрации широковещательного приемника, отслеживающего состояние сертификатов в системе.

21. public void handleProfile(ProfileResponse profileResponse) – метод, в котором обрабатывается полученный с сервера профиль. Параметр profileResponse – объект типа ProfileResponse. Представляет собой профиль, полученный в ответе на запрос к серверу.

22. private void handleUserCertificate(TrustedCertificateEntry entry) – метод, в котором обрабатывается сертификат. Параметр entry – объект типа TrustedCertificateEntry. Представляет собой сертификат, который загружается из системы.

23. private void saveProfile() – метод, в котором происходит сохранение профиля в систему.

24. private boolean isServiceConnected() – метод, который возвращает значение true или false в зависимости от того активен ли VPN сервис.

25. public void startVpnProfile(Intent intent, boolean foreground) – метод, который запускает VPN сервис. Параметры: intent – объект типа Intent, который вызвал запуск сервиса, foreground – значение типа boolean, которое означает был ли вызван запуск когда приложение было активно.

26. public void startVpnProfile(VpnProfile profile, boolean foreground) – метод, который запускает VPN сервис. Параметры: profile – объект типа VpnProfile, который представляет собой профиль, с которым необходимо запустить VPN сервис, foreground – значение типа boolean, которое означает был ли вызван запуск когда приложение было активно.

27. protected void prepareVpnService(Bundle profileInfo) – метод, который подготавливает VPN сервис. Если все проходит успешно, то VPN сервис запускается. Параметр profileInfo – объект типа Bundle, который хранит информацию о том, какой профиль необходимо запустить.

28. public void disconnect() – метод, который останавливает текущий VPN сервис.

**3.1.9** Класс ReportFragment

Класс, представляющий собой отображение экрана статистики.

Поля:

– mReportResponse – поле типа ReportReponse, представляющее из себя ответ сервера на запрос по статистике;

– mCounter – поле типа int, представляющее из себя счетчик запросов;

– OTHER\_ENTRY – константное поле типа String, представляющее из себя лейбл для графика;

– mScreenWidthPx – поле типа int, представляющее из себя ширину экрана;

– mFromTime – поле типа String, представляющее из себя начальную дату для выдачи статистики;

– mToTime – поле типа String, представляющее из себя конечную дату для выдачи статистики.

Методы:

1. protected int getLayoutId() – переопределенный базовый метод, который возвращает значение идентификатора файла разметки экрана.

2. protected void initViews() – реализованный метод базового класса BaseFragment, в котором инициализируются все UI компоненты.

3. private int getLastTabIndex() – метод, который индекс последнего выбранного таба.

4. private void setLastTabIndex(final int index) – метод, который устанавливает значение индекса последнего выбранного таба. Параметр index – значение типа int, которое необходимо установить.

5. private int initCharts() – метод, в котором происходит инициализация графиков.

6. private int requestData() – метод, в котором отправляется запрос на получение статистики за выбранный период.

7. private int reportRequest(int count, String tabText) – метод, в котором отправляется запрос на получение конкретного вида статистики за выбранный период. Параметры: count – значение типа int, которое означает какой вид статистики необходимо получить, tabText – значение типа String, которое означает за какой период необходимо получить статистику.

8. private void drawCahrt(int counter, Report report) – метод, в котором рисуется график по конкретному виду статистики. Параметры: counter – значение типа int, которое означает какой график необходимо рисовать, report – объект типа Report, который хранит значения для построения графика.

9. private int clearCharts() – метод, в котором происходит очистка всех графиков.

**3.1.10** Класс ApiPath

Класс представляет из себя набор константных статических переменных, хранящих URL для запросов.

Поля:

– VPN\_PROFILE – статическое константное поле типа String, хранящее URL для получения VPN профиля;

– DATA\_SERVICE\_LIST – статическое константное поле типа String, хранящее URL для получения всех видов статистики;

– DATA\_REPORT – статическое константное поле типа String, хранящее URL для получения данных по конкретному виду статистики.

**3.1.11** Класс ApiManager

Класс представляет из себя набор полей и методов отвечающих за отправку запросов к серверу и получение ответов от сервера.

Поля:

– TIME\_OUT – статическое константное поле типа int, хранящее значение ожидания запроса;

– mServiceApi – константное поле типа IServiceApi, являющееся объектом для получения доступа к методам для отправки запроса.

Методы:

1. public static ApiManager getNewInstance() – статический метод, который возвращает новый объект класса ApiManager.

2. private OkHttpClient buildHttpClient() – метод, который создает Http клиент для отправки запросов и получения ответов от сервера.

3. public IServiceApi getServiceApi() – метод, который возвращает объект mServiceApi.

4. private void disableSSLCertificateChecking(final OkHttpClient.Builder builder) – метод, который отключает проверку SSL сертификатов при запросе для Http клиента. Параметр builder – объект типа OkHttpClient.Builder представляет собой клиент, для которого необходимо отключить проверку сертификатов.

**3.1.12** Интерфейс IServiceApi

Интерфейс представляет из себя набор методов, с помощью которых будет осуществляться запрос. Каждый метод помечается аннотацией @GET или @POST, в которую передается URL для запроса. Также аннотацией @Headers перечисляются заголовки запроса.

Методы:

1. LiveData<ApiResponse<ServiceListReponse>> getServiceList() – метод получения списка доступных видов статистики.

2. LiveData<ApiResponse<ProfileReponse>> getVpnProfile(@Body ProfileRequest data) – метод получения VPN профиля. Параметр data является объектом типа ProfileRequest и представляет из себя тело запроса. Аннотация @Body означает, что параметр является телом запроса.

3. LiveData<ApiResponse<ReportReponse>> getReportData(@Body ReportRequest data) – метод получения данных по конкретному виду статистики. Параметр data является объектом типа ReportRequest и представляет из себя тело запроса. Аннотация @Body означает, что параметр является телом запроса.

**3.1.13** Класс BasePreferencesManager

Базовый класс, который представляет набор методов для работы с SharedPreferences.

Поля:

– mSettings – константное поле типа SharedPreferences, являющееся объектом для доступа к файлу общих настроек приложения;

– mGson – константное поле типа Gson, являющееся объектом для конвертации объектов в строки формата JSON.

Методы:

1. void setString(final String key, final String value) – метод сохраняет значение типа String в файл общих настроек. Параметры: key – значение типа String, которое является ключом к сохраняемой строке, value – значение типа String, которое необходимо сохранить.

2. void setBoolean(final String key, final boolean value) – метод сохраняет значение типа boolean в файл общих настроек. Параметры: key – значение типа String, которое является ключом к сохраняемой строке, value – значение типа boolean, которое необходимо сохранить.

3. void setLong(final String key, final long value) – метод сохраняет значение типа long в файл общих настроек. Параметры: key – значение типа String, которое является ключом к сохраняемой строке, value – значение типа long, которое необходимо сохранить.

4. void getString(final String key, final String defaultValue) – метод получения значения типа String из файл общих настроек. Параметры: key – значение типа String, которое является ключом для получения значения, defaultValue – значение типа String, которое возвращается в случае, если по данному ключу значения нет.

5. void getBoolean(final String key, final boolean defaultValue) – метод получения значения типа boolean из файл общих настроек. Параметры: key – значение типа String, которое является ключом для получения значения, defaultValue – значение типа boolean, которое возвращается в случае, если по данному ключу значения нет.

6. void getLong(final String key, final long defaultValue) – метод получения значения типа long из файл общих настроек. Параметры: key – значение типа String, которое является ключом для получения значения, defaultValue – значение типа long, которое возвращается в случае, если по данному ключу значения нет.

7. void drop(final String key) – метод удаления значения из файл общих настроек по ключу. Параметр key – значение типа String, является ключом, по которому необходимо удалить значение.

**3.1.14** Класс PreferencesManager

Класс является наследником BasePreferencesManager. В классе имеются поля, которые являются ключами к значениям в файле общих настроек. Также определены методы с более понятными для работы названиями для доступа к значениям в файле общих настроек.

Поля:

– PREFS\_FILE\_NAME – статическое константное поле типа String, хранящее имя файла общих настроек;

– PREFS\_UUID – статическое константное поле типа String, являющееся ключом для получения и записи уникального идентификатора пользователя;

– PREFS\_SESSION\_START\_TIME – статическое константное поле типа String, являющееся ключом для получения и записи времени начала сессии;

– PREFS\_IS\_TREATS\_BLOCKED – статическое константное поле типа String, являющееся ключом для получения и записи состоянии о заблокированных угрозах;

– PREFS\_IS\_CERTIFICATE\_INSTALLED – статическое константное поле типа String, являющееся ключом для получения и записи состояния установки сертификата;

– PREFS\_IS\_NEED\_TO\_SCROLL – статическое константное поле типа String, являющееся ключом для получения и записи значения о необходимости прокручивать экран;

– PREFS\_IS\_VPN\_CONNECTED – статическое константное поле типа String, являющееся ключом для получения и записи значения состояния VPN соединения.

Методы:

1. public String getUUID() – метод получения значения уникального идентификатора пользователя из файла общих настроек.

2. public void setUUID(final String uuid) – метод записи значения уникального идентификатора пользователя в файл общих настроек. Параметр uuid – значение типа String, которое записывается в файл общих настроек.

3. public String getUserName() – метод получения значения уникального имени пользователя из файла конфигурации приложения.

4. public String getUserPassword() – метод получения значения пароля пользователя из файла конфигурации приложения.

5. public boolean getIsVpnConnected() – метод получения значения состояния VPN соединения.

6. public void setIsVpnConnected(final boolean isVpnConnected) – метод записи значения состояния VPN соединения в файл общих настроек. Параметр isVpnConnected – значение типа boolean, которое записывается в файл общих настроек.

7. public long getSessionStartTime() – метод получения значения времени начала сессии.

8. public void setSessionStartTime(final long time) – метод записи значения времени начала сессии в файл общих настроек. Параметр time – значение типа long, которое записывается в файл общих настроек.

9. public void clearSessionStartTime() – метод удаления значения времени начала сессии из файла общих настроек.

10. public void setisTreatsBlocked(final boolean isTreatsBlocked) – метод записи значения состояния о заблокированных угрозах в файл общих настроек. Параметр isTreatsBlocked – значение типа boolean, которое записывается в файл общих настроек.

11. public boolean isTreatsBlocked() – метод получения значения состояния о заблокированных угрозах из файла общих настроек.

12. public void setIsCertInstalled(final boolean isInstalled) – метод записи значения состояния установки сертификата в файл общих настроек. Параметр isInstalled – значение типа boolean, которое записывается в файл общих настроек.

13. public boolean isCertInstalled() – метод получения значения состояния установки сертификата из файла общих настроек.

14. public void setIsNeedToScroll(final boolean isNeedToScroll) – метод записи значения о необходимости прокручивания экрана в файл общих настроек. Параметр isNeedToScroll – значение типа boolean, которое записывается в файл общих настроек.

15. public boolean isNeedToScroll() – метод получения значения о необходимости прокручивания экрана из файла общих настроек.

16. public String getAuthorizationValue() – метод получения значения для авторизации.

**3.1.15** Класс TrustedCertificateEntry

Класс представляет из себя обертку для сертификата в коде программы.

Поля:

– mCert – константное поле типа X509Certificate представляет собой сертификат;

– alias – константное поле типа String, являющееся именем сертификата;

– mSubjectPrimary – поле типа String представляет основное назначение данного сертификата;

– mSubjectSecondary – поле типа String представляет вторичное назначение данного сертификата.

Методы:

1. public String getAlias() – метод возвращает имя сертификата.

2. public X509Certificate getCertificate() – метод возвращает сертификат.

**3.1.16** Класс BootCompletedReceiver

Класс представляет реализацию широковещательного приемника, отслеживающего перезагрузку устройства.

Поля:

– BOOT\_COMLETED – статическое константное поле типа String, являющееся действием окончания загрузки устройства;

– REBOOT – статическое константное поле типа String, являющееся действием окончания перезагрузки устройства.

Методы:

1. public void onReceive(Context context, Intent intent) – метод, вызывающийся когда широковещательный приемник перехватывает действие. В качестве входного параметра принимает переменную типа Context, представляющую собой контекст, и Intent, представляющую собой объект, который вызвал данное действие.

2. private void restartVpn(Context context) – метод перезапускающий VPN после перезагрузки устройства. В качестве входного параметра принимает переменную типа Context, представляющую собой контекст.

**3.1.17** Класс ConnectivityUtils

Класс представляет из себя набор статических методов по преобразованию данных из одного типа в другой.

Поля:

– instance – статическое поле типа ConnectivtyUtils, являющееся сущностью класса ConnectivtyUtils;

– mConnectListener – поле типа ConnectivityInterface, являющееся интерфейсом, предоставляющим метод для вызова при смене состояния соединения;

– mConnectReceiver – поле типа ConnectivityReceiver, являющееся объектом широковещательного приемника, отслеживающего состояние соединения.

Методы:

1. public static ConnectivtyUtils getInstance() – статический метод получения сущности класса ConnectivtyUtils.

2. public ConnectivityReceiver getConnectReceiver(final ConnectivityInterface connectListener) – метод получения широковещательного приемника, отслеживающего состояние соединения. В качестве входного параметра принимает переменную типа ConnectivityInterface, представляющую собой метод обратного вызова, который срабатывает при изменении состояния соединения.

3. public static boolean isConnected(final Context context) – метод возвращает текущее состояние соединения. В качестве входного параметра принимает переменную типа Context, представляющую собой контекст.

**3.1.18** Класс EncodeUtil

Класс содержит в себе статический метод дешифрования из формата Base64.

Методы:

1. public static byte[] decodeBase64(String encoded) – метод преобразования зашифрованного текста в массив байт. В качестве входного параметра принимает переменную типа String, представляющую собой текст для дешифрования.

**3.1.19** Класс CharonVpnService

Класс представляет из себя реализацию сервиса, который управляет VPN соединением.

Поля:

– VPN\_STATE\_NOTIFICATION\_ID – константное статическое поле типа int, являющееся идентификатором уведомления данного сервиса;

– mConnectionHandler – поле типа Thread, являющееся потоком, в котором выполняется вся работа VPN;

– mCurrentProfile – поле типа VpnProfile, являющееся объектом текущего VPN профиля;

– mCurrentCertificateAlias – поле типа String, являющееся именем текущего сертификата;

– mCurrentUserCertificateAlias – поле типа String, являющееся именем текущего сертификата пользователя;

– mNextProfile – поле типа VpnProfile, являющееся объектом следующего VPN профиля, который необходимо запустить;

– mProfileUpdated – поле типа boolean, являющееся флагом того, что профиль обновился;

– mTerminate – поле типа boolean, являющееся флагом того, что соединение завершено;

– mIsDisconnecting – поле типа boolean, являющееся флагом того, что соединение разрывается;

– mShowNorification – поле типа boolean, являющееся флагом того, что уведомление отображается;

– mService – поле типа VpnStateService, являющееся объектом сервиса, отслеживающего состояние VPN соединения;

– mServiceConnection – поле типа ServiceConnection, являющееся сервисом, отслеживающим состояние текущего сервиса;

– mCurrentProfile – поле типа VpnProfile, являющееся объектом текущего VPN профиля.

Методы:

1. public void onCreate() – метод жизненного цикла сервиса. Вызывается при создании сервиса.

2. public void onRevoke() – метод отключение сервиса. Вызывается, когда пользователь отключает VPN через настройки.

3. public void onDestroy() – метод жизненного цикла сервиса. Вызывается при завершении сервиса.

4. public void setNextProfile(VpnProfile profile) – метод устанавливает профиль, который будет инициализирован. Профиль передается в качестве параметра profile.

5. public void run() – метод, в котором происходит основная работа VPN. Вызывается в потоке mConnectionHandler.

6. private void stopCurrentConnection() – метод останавливает текущее VPN соединение.

8. private void addNotification() – метод добавляет уведомление с состоянием VPN соединения.

9. private void removeNotification() – метод удаляет уведомление с состоянием VPN соединения.

10. private Notification buildNotification(boolean publicVersion) – метод создает уведомление с текущим состоянием VPN. В качестве параметра принимает переменную publicVersion, которая является флагом того, необходимо ли отображать содержимое на заблокированном экране.

11. private void startConnection(VpnProfile profile) – метод уведомляет сервис, который отслеживает состояние VPN, о том, что создается новое соединение. В качестве параметра принимает переменную profile, которая является активным VPN профилем.

12. private void setState(State state) – метод обновляет текущее состояние в сервисе, который отслеживает состояние VPN. В качестве параметра принимает переменную state, которая является текущим состоянием.

13. private void setError(ErrorState error) – метод обновляет текущее состояние ошибки в сервисе, который отслеживает состояние VPN. В качестве параметра принимает переменную error, которая является текущим состоянием ошибки.

**3.1.20** Класс TrustedCertificateManager

В данном классе реализован функционал по извлечению всех сертификатов или из конкретного хранилища.

Поля:

– mLock – константное поле типа ReentrantReadWriteLock, являющееся защелкой при чтении и записи;

– mCACerts – поле типа HashTable<String, X509Certificate>, являющееся хэш-таблицей, в которой хранятся сертификаты;

– mReload – поле типа boolean, являющееся флагом того, что была вызвана перезагрузка сертификатов;

– mLoaded – поле типа boolean, являющееся флагом того, что сертификаты были загружены;

– mKeyStores – поле типа ArrayList<KeyStore>, являющееся списком источников для извлечения сертификатов.

Методы:

1. public static TrustedCertificateManager getInstance() – метод возвращает сущность класса TrustedCertificateManager.

2. public TrustedCertificateManager reset() – метод перезагружает кэшированные сертификаты.

3. public TrustedCertificateManager load() – метод проверяет загружены ли сертификаты. Если не загружены, то начинается загрузка сертификатов.

4. private void loadCertificates() – метод открывает хранилище сертификатов и начинает загрузку кэшированных сертификатов.

5. private void fetchCertificates(Hashtable<String, X509Certificate> certs, KeyStore store) – метод загружает все сертификаты из хранилища. Хранилище, из которого необходим выгружать, передается в качестве параметра store. Сертификаты загружаются в хэш-таблицу, которая передается в качестве параметра certs.

6. private X509Certificate getX509CertificateFromAlias(String alias) – метод возвращает сертификат по имени, которое передается в качестве параметра alias.

7. private Hashtable<String, X509Certificate> getAllCertificates() – метод возвращает все сертификаты из всех хранилищ.

Перечисления:

– TrustedCertificateSource – перечисление источников доверенных сертификатов.

**3.1.21** Класс VpnStateService

Класс представляет из себя реализацию сервиса, который отслеживает состояние VPN соединения.

Поля:

– mListeners – поле типа HashSet<VpnStateListener>, являющееся хэш-набором, который хранит всех слушателей, отслеживающих состояние VPN соединения;

– mConnectionID – поле типа long, являющееся идентификатором текущего VPN соединения;

– mHandler – поле типа Handler, являющееся объектом, который будет оповещать всех слушателей о том, что состояние VPN соединения изменилось;

– mProfile – поле типа VpnProfile, являющееся флагом того, что сертификаты были загружены;

– mState – поле типа State, являющееся текущим состоянием VPN соединения;

– mError – поле типа ErrorState, являющееся текущим состоянием ошибки VPN соединения.

Методы:

1. public void onCreate() – метод жизненного цикла сервиса. Вызывается при создании сервиса.

2. public void onDestroy() – метод жизненного цикла сервиса.

Вызывается при завершении сервиса.

3. public void registerListener(VpnStateListener listener) – метод регистрирует нового слушателя, отслеживающего изменения состояния VPN соединения. Новый слушатель передается в качестве параметра listener.

4. public void unRegisterListener(VpnStateListener listener) – метод отменяет регистрацию слушателя, который передается в качестве параметра listener.

5. public VpnProfile getProfile() – метод возвращает текущий VPN профиль.

6. public State getState() – метод возвращает текущее состояние VPN соединения.

7. public ErrorState getErrorState() – метод возвращает текущее состояние ошибки VPN соединения.

8. public void disconnect() – метод останавливает текущее соединение. При этом VPN сервис не останавливается, а сбрасывается. Таким образом новое соединение может быть добавлено.

9. public void notifyListeners(final Callable<Boolean> change) – метод уведомляет всех слушателей о том, что состояние VPN соединения изменилось. Если действие, которое передается в качестве параметра change, не удалось выполнить, то слушатели уведомляться не будут.

10. public void startConnection(final VpnProfile profile) – метод вызывается когда создается новое соединение. Устанавливается новый профиль, который передается в качестве параметра profile. Также сбрасывается состояние ошибки VPN соединения, состояние устанавливается в значение CONNECTING, увеличивается идентификатор текущего VPN соединения и оповещаются все слушатели.

11. public void setState(final State state) – метод устанавливает текущее состояние VPN соединения. Состояние, в которое необходимо установить, передается в качестве параметра state.

12. public void setErrorState(final ErrorState error) – метод устанавливает текущее состояние ошибки VPN соединения. Состояние ошибки, в которое необходимо установить, передается в качестве параметра error.

Перечисления:

– State – перечисление состояний VPN соединения;

– ErrorState – перечисление состояний ошибки VPN соединения.

**3.1.22** Перечисление VpnType

Перечисление представляет из себя набор типов VPN. Каждый VPN типа характеризуется идентификатором и набором функций, которые передаются в качестве параметров для каждого типа.

Поля:

– IKEV2\_EAP;

– IKEV2\_CERT;

– IKEV2\_CERT\_EAP;

– IKEV2\_EAP\_TLS;

– IKEV2\_BYOD\_EAP.

Методы:

1. public String getIdentifier() – метод возвращает идентификатор VPN типа.

2. public boolean has(VpnTypeFeature feature) – метод проверяет имеет ли данный VPN типа функцию. Функция передается в качестве параметра feature.

3. public VpnType fromIdentifier(String identifier) – статический метод получения VPN типа по идентификатору. В качестве параметра передается переменная identifier, по которой необходимо определить VPN тип.

Перечисления:

– VpnTypeFeature – перечисление функций для VPN типа.