Package: firebase (via r-universe)

October 23, 2024

Title Integrates 'Google Firebase' Authentication Storage, and 'Analytics' with 'Shiny'

Version 1.0.2

Date 2023-07-07

Description Authenticate users in 'Shiny' applications using 'Google Firebase' with any of the many methods provided; email and password, email link, or using a third-party provider such as 'Github', 'Twitter', or 'Google'. Use 'Firebase Storage' to store files securely, and leverage 'Firebase Analytics' to easily log events and better understand your audience.

License GPL (>= 3)

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

Imports cli, jose, shiny, openssl, jsonlite, base64enc, htmltools

Suggests R6 (>= 2.5.0)

RoxygenNote 7.2.3

URL https://firebase.john-coene.com/, https://github.com/JohnCoene/firebase

BugReports https://github.com/JohnCoene/firebase/issues

Repository https://johncoene.r-universe.dev

RemoteUrl https://github.com/johncoene/firebase

RemoteRef HEAD

RemoteSha 01bf3e4f1186aa1bcb6c106eac1b55fc713d9d49

Contents

`Analytics` *Analytics*

Description

Analytics

Analytics

Value

An object of class `Analytics`.

Super class

`firebase::Firebase` -> `Analytics`

Methods

Public methods:

- `Analytics$new()`
- `Analytics$launch()`
- `Analytics$enable()`
- `Analytics$disable()`
- `Analytics$log_event()`
- `Analytics$set_user_properties()`
- `Analytics$clone()`

Method `new()`:

Usage:

```
Analytics$new(enable = TRUE)
```

Arguments:

`enable` Whether to internally enable analytics see `launch` method.

Details: Initialise
Initialise an analytics object.

Method `launch()`**:**

Usage:

```
Analytics$launch()
```

Details: Launch
Launch the analytics tracking. Note that analytics is not launched by the constructor in order to be able to enable or disable the tracking prior to the launch. This is because once Google Analytics is launched it cannot be disabled. If needed ask the user before running this method. The enabling and disabling of tracking provided by the package is only internal, e.g.: disabling tracking during a session will stop the `log_event` method from registering event but default Google Analytics will still be running.

Method `enable()`**:**

Usage:

```
Analytics$enable()
```

Details: Enable Tracking Internally enables tracking.

Method `disable()`**:**

Usage:

```
Analytics$disable()
```

Details: Disable Tracking Internally disables tracking: running methods from this instance of the class will not actually register with Google Analytics.

Method `log_event()`**:**

Usage:

```
Analytics$log_event(event, params = NULL)
```

Arguments:

`event` Event to log choose from this list of supported events.

`params` Event parameters.

Details: Log Event
Log an event.

Method `set_user_properties()`**:**

Usage:

```
Analytics$set_user_properties(...)
```

Arguments:

`...` Named arguments defining the properties of the user.

Details: Set user properties

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Analytics$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`config` *Config*

Description

Configure Firebase, either using a config file or by setting environment variables (see section below).

Usage

```
firebase_config(
  api_key,
  project_id,
  auth_domain = NULL,
  storage_bucket = NULL,
  app_id = NULL,
  database_url = NULL,
  overwrite = FALSE
)
```

Arguments

`api_key`	API key of your project.
`project_id`	Id of your web project.
`auth_domain`	Authentication domain, if `NULL` attempts to build firebase's default domain.
`storage_bucket`	URl to the bucket. if `NULL` attempts to build firebase's default storage domain.
`app_id`	Application ID, necessary for Analytics.
`database_url`	URL to the database, required to use the `RealtimeDatabase`.
`overwrite`	Whether to overwrite any existing configuration file.

Details

Creates the configuration file necessary to running fireblaze. Note that if you changed the project you must use said ID here, not the one originally created by Google.

Classes of the package look first for the configuration file then, if not found look for the environment variables.

Value

Path to file.

Environment Variables

- FIREBASE_API_KEY
- FIREBASE_PROJECT_ID
- FIREBASE_AUTH_DOMAIN
- FIREBASE_STORAGE_BUCKET
- FIREBASE_APP_ID
- FIREBASE_DATABASE_URL

Note

Do not share this file with anyone.

Examples

```
## Not run: firebase_config("xXxxx", "my-project")
```

---

dependencies *Dependencies*

Description

Include dependencies in your Shiny application. `use_firebase` *must* be included in every application.

Usage

```
useFirebase(analytics = FALSE, firestore = FALSE)

useFirebaseUI(...)

firebaseUIContainer()
```

Arguments

analytics	Deprecated. Whether to include analytics.
firestore	Whether to include firestore.
...	Ignored, for backwards compatibility.

Details

Place `useFirebaseUI` *where* you want the pre-built UI to be placed. Otherwise one

Value

No return value, called for side effects.

Functions

- `useFirebase` Is required for every app that uses this package
- `useFirebaseUI` Is required for applications that use `FirebaseUI`
- `firebaseUIContainer` To place the container of the pre-built UI where desired

---

`Firebase` *Firebase*

Description

Core Firebase class.

Value

An object of class `Firebase`.

Public fields

`session` A valid Shiny session.

Methods

Public methods:

- `Firebase$new()`
- `Firebase$expose_app()`
- `Firebase$print()`
- `Firebase$clone()`

Method `new()`:

Usage:

```
Firebase$new(
  config_path = "firebase.rds",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`config_path` Path to the configuration file as created by `firebase_config`.

`session` A valid shiny session.

Details: Initialise Firebase
Initialises the Firebase application client-side.

Returns: Invisibly return the class.

Method `expose_app()`:

Usage:

```
Firebase$expose_app()
```

Details: Expose App
Expose the `firebaseApp` object product of `initializeApp()` by attaching it to the `window`: access it with `window.firebaseApp`.

Method `print()`**:**

Usage:

```
Firebase$print()
```

Details: Print the class

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Firebase$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FirebaseAuth` *Firebase Authentication*

Description

Use firebase to manage authentications.

Value

An object of class `FirebaseAuth`.

Super class

`firebase::Firebase` -> `FirebaseAuth`

Active bindings

`signed_in` Read the signed in user.

`signed_up` Read the signed in user.

Methods

Public methods:

- `FirebaseAuth$new()`
- `FirebaseAuth$print()`
- `FirebaseAuth$sign_out()`
- `FirebaseAuth$get_sign_out()`
- `FirebaseAuth$get_signed_in()`
- `FirebaseAuth$get_signed_up()`
- `FirebaseAuth$is_signed_in()`

- FirebaseAuth$req_sign_in()
- FirebaseAuth$req_sign_out()
- FirebaseAuth$set_language_code()
- FirebaseAuth$delete_user()
- FirebaseAuth$get_delete_user()
- FirebaseAuth$expose_auth()
- FirebaseAuth$get_access_token()
- FirebaseAuth$clear()
- FirebaseAuth$request_id_token()
- FirebaseAuth$get_id_token()
- FirebaseAuth$clone()

Method `new()`**:**

Usage:

```
FirebaseAuth$new(
  persistence = c("session", "local", "memory"),
  config_path = "firebase.rds",
  language_code = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`persistence` How the auth should persit: `none`, the user has to sign in at every visit, `session` will only persist in current tab, `local` persist even when window is closed.

`config_path` Path to the configuration file as created by `firebase_config`.

`language_code` Sets the language to use for the UI. Supported languages are listed here. Set to `browser` to use the default browser language of the user.

`session` A valid shiny session.

Details: Initialise firebase authentication

Method `print()`**:**

Usage:

```
FirebaseAuth$print()
```

Details: Print the class

Method `sign_out()`**:**

Usage:

```
FirebaseAuth$sign_out()
```

Details: Signs out user

Returns: self

Method `get_sign_out()`**:**

Usage:

```
FirebaseAuth$get_sign_out()
```

Details: Get signed out results

Returns: A list of length 2 containing `success` a boolean indicating wherther signing out was successful and `response` containing `sucessful` or the error.

Method `get_signed_in()`**:**

Usage:

```
FirebaseAuth$get_signed_in()
```

Details: Signed in user details triggered when auth states changes

Returns: A list of length 2 containing `success` a boolean indicating wherther signing in was successful and `response` containing the user object or `NULL` if signing in failed.

Method `get_signed_up()`**:**

Usage:

```
FirebaseAuth$get_signed_up()
```

Details: Get results of a sign up

Returns: A list of length 2 containing `success` a boolean indicating wherther signing in was successful and `response` containing the user object or `NULL` if signing in failed.

Method `is_signed_in()`**:**

Usage:

```
FirebaseAuth$is_signed_in()
```

Details: Check whether use is signed in

Returns: A boolean indicating whether user has successfully signed in.

Method `req_sign_in()`**:**

Usage:

```
FirebaseAuth$req_sign_in()
```

Details: Makes Shiny output, observer, or reactive require the user to be signed in

Method `req_sign_out()`**:**

Usage:

```
FirebaseAuth$req_sign_out()
```

Details: Makes Shiny output, observer, or reactive require the user to be signed out

Method `set_language_code()`**:**

Usage:

```
FirebaseAuth$set_language_code(code)
```

Arguments:

`code` iso639-1 language code.

Details: Set language code for auth provider

Returns: self

Method `delete_user()`**:**

Usage:

```
FirebaseAuth$delete_user()
```

Details: Delete the user

Returns: self

Method `get_delete_user()`**:**

Usage:

```
FirebaseAuth$get_delete_user()
```

Details: Get result of user deletion

Returns: A list of length 2 containing `success` a boolean indicating wherther deletion was successful and `response` containing either `successful` string or the error if signing in failed.

Method `expose_auth()`**:**

Usage:

```
FirebaseAuth$expose_auth()
```

Details: Expose Auth
Expose the `firebaseAuth` object the product of the authentication attaching it to the `window`: access it with `window.firebaseAuth`.

Method `get_access_token()`**:**

Usage:

```
FirebaseAuth$get_access_token()
```

Details: Get user access token

Returns: User's access token

Method `clear()`**:**

Usage:

```
FirebaseAuth$clear()
```

Details: Clear user session
This clears the login internally and will retrigger a JWT token check, only useful if you are running really long sessions.

Method `request_id_token()`**:**

Usage:

```
FirebaseAuth$request_id_token()
```

Details: Request the users' ID Token
Used to retrieved the user's ID token useful to connect with other Google APIs and make request on the user's behalf. This executes the request for the id token, this request can only be made once the user is signed in.
The actual id token is obtained with the `get_id_token` method.

Method `get_id_token()`**:**

Usage:

```
FirebaseAuth$get_id_token()
```

Details: Retrieve the users' ID Token
Also see `request_id_token`.

Returns: the id token (invisibly).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FirebaseAuth$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FirebaseEmailLink` *Email Link*

Description

Sign in the user by emailing them a link.

Value

An object of class `FirebaseEmailLink`.

Super classes

`firebase::Firebase` -> `firebase::FirebaseAuth` -> `FirebaseEmailLink`

Active bindings

`email_verification` Email verification results

`email_sent` Email send results

Methods

Public methods:

- `FirebaseEmailLink$new()`
- `FirebaseEmailLink$config()`
- `FirebaseEmailLink$send_email()`
- `FirebaseEmailLink$get_email_sent()`
- `FirebaseEmailLink$get_email_verification()`
- `FirebaseEmailLink$clone()`

Method `new()`**:**

Usage:

```
FirebaseEmailLink$new(
  persistence = c("session", "local", "memory"),
  config_path = "firebase.rds",
  language_code = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`persistence` How the auth should persit: `none`, the user has to sign in at every visit, `session` will only persist in current tab, `local` persist even when window is closed.

`config_path` Path to the configuration file as created by `firebase_config`.

`language_code` Sets the language to use for the UI. Supported languages are listed here. Set to `browser` to use the default browser language of the user.

`session` A valid shiny session.

Details: Initialiases Firebase Email Link
Initialises the Firebase application client-side.

Method `config()`**:**

Usage:

```
FirebaseEmailLink$config(url, ...)
```

Arguments:

`url` The link is handled in the web action widgets, this is the deep link in the continueUrl query parameter. Likely, your shiny application link.

`...` Any other parameter from the official documentation.

Details: Configure

Examples:

```
\dontrun{
f <- FirebaseEmailLink$
 new()$ # create
 config(url = "https://me.shinyapps.io/myApp/")
}
```

Method `send_email()`**:**

Usage:

```
FirebaseEmailLink$send_email(email)
```

Arguments:

`email` Email to send verification to.

Details: Send email verification link.

Returns: self

Examples:

```
\dontrun{
f <- FirebaseEmailLink$
 new()$ # create
 config(url = "https://me.shinyapps.io/myApp/")$
 send("user@email.com")
}
```

Method `get_email_sent()`**:**

Usage:

```
FirebaseEmailLink$get_email_sent()
```

Details: Get whether email verification was correctly sent.

Returns: A list of length 2 containing `success` a boolean indicating wherther sending the email was successful and `response` containing the email used to sign in or the error if sending failed.

Method `get_email_verification()`**:**

Usage:

```
FirebaseEmailLink$get_email_verification()
```

Details: Get whether user is signing in from email verification.

Returns: A list of length 2 containing `success` a boolean indicating wherther signing in from the verification link was successful and `response` containing the result of the sign in or the error if signing in failed.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FirebaseEmailLink$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

Other methods to pick up whether user signs in still apply. This is for added security measures.

Examples

```
library(shiny)
library(firebase)
old <- options()
options(shiny.port = 3000)

ui <- fluidPage(
  useFirebase(),
  textInput("email", "Your email"),
  actionButton("submit", "Submit")
)

server <- function(input, output){

  f <- FirebaseEmailLink$
    new()$
    config(url = "http://127.0.0.1:3000")

  observeEvent(input$submit, {
    if(input$email == "")
      return()
```

```
    f$send(input$email)
  })

  observeEvent(f$get_email_sent(), {
    sent <- f$get_email_sent()

    if(sent$success)
      showNotification("Email sent", type = "message")
  })

  observeEvent(f$get_email_verification(), {
    print(f$get_email_verification())
  })

}

if(interactive()){
   shinyApp(ui, server)
}


options(old)


## ------------------------------------------------
## Method `FirebaseEmailLink$config`
## ------------------------------------------------

## Not run:
f <- FirebaseEmailLink$
 new()$ # create
 config(url = "https://me.shinyapps.io/myApp/")

## End(Not run)

## ------------------------------------------------
## Method `FirebaseEmailLink$send_email`
## ------------------------------------------------

## Not run:
f <- FirebaseEmailLink$
 new()$ # create
 config(url = "https://me.shinyapps.io/myApp/")$
 send("user@email.com")

## End(Not run)
```

FirebaseEmailPassword *Email & Password*

Description

Manage users using email and password.

Value

An object of class `FirebaseEmailPassword`.

Super classes

`firebase::Firebase` -> `firebase::FirebaseAuth` -> `FirebaseEmailPassword`

Active bindings

`created` Results of account creation

Methods

Public methods:

- `FirebaseEmailPassword$new()`
- `FirebaseEmailPassword$create()`
- `FirebaseEmailPassword$sign_in()`
- `FirebaseEmailPassword$get_created()`
- `FirebaseEmailPassword$reset_password()`
- `FirebaseEmailPassword$get_reset()`
- `FirebaseEmailPassword$send_verification_email()`
- `FirebaseEmailPassword$get_verification_email()`
- `FirebaseEmailPassword$set_password()`
- `FirebaseEmailPassword$get_password()`
- `FirebaseEmailPassword$re_authenticate()`
- `FirebaseEmailPassword$get_re_authenticated()`
- `FirebaseEmailPassword$clone()`

Method `new()`:

Usage:

```
FirebaseEmailPassword$new(
  persistence = c("session", "local", "memory"),
  config_path = "firebase.rds",
  language_code = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`persistence` How the auth should persit: `none`, the user has to sign in at every visit, `session` will only persist in current tab, `local` persist even when window is closed.

`config_path` Path to the configuration file as created by `firebase_config`.

`language_code` Sets the language to use for the UI. Supported languages are listed here. Set to `browser` to use the default browser language of the user.

`session` A valid shiny session.

Details: Initialiases Firebase Email Password
Initialises the Firebase application client-side.

Method `create()`**:**

Usage:

```
FirebaseEmailPassword$create(email, password)
```

Arguments:

`email, password` Credentials as entered by the user.

Details: Create an account

Returns: self

Method `sign_in()`**:**

Usage:

```
FirebaseEmailPassword$sign_in(email, password)
```

Arguments:

`email, password` Credentials as entered by the user.

Details: Sign in with email

Returns: `NULL` if successful, the error otherwise.

Method `get_created()`**:**

Usage:

```
FirebaseEmailPassword$get_created()
```

Details: Get account creation results

Returns: A list of length 2 containing `success` a boolean indicating wherther creation was successful and `response` containing the result of account creation or the error if failed.

Method `reset_password()`**:**

Usage:

```
FirebaseEmailPassword$reset_password(email = NULL)
```

Arguments:

`email` Email to send reset link to, if missing looks for current logged in user's email.

Details: Reset user password

Returns: self

Method `get_reset()`**:**

Usage:

```
FirebaseEmailPassword$get_reset()
```

Details: Get whether password reset email was successfully sent

Returns: A list of length 2 containing `success` a boolean indicating whether email reset was successful and `response` containing `successful` or the error.

Method `send_verification_email()`**:**

Usage:

```
FirebaseEmailPassword$send_verification_email()
```

Details: Send the user a verification email

Returns: self

Method `get_verification_email()`**:**

Usage:

```
FirebaseEmailPassword$get_verification_email()
```

Details: Get result of verification email sending procedure

Returns: A list of length 2 containing `success` a boolean indicating whether email verification was successfully sent and `response` containing `successful` or the error.

Method `set_password()`**:**

Usage:

```
FirebaseEmailPassword$set_password(password)
```

Arguments:

`password` The authenticated user password, the user should be prompted to enter it.

Details: Set user password
Useful to provide ability to change password.

Returns: self

Method `get_password()`**:**

Usage:

```
FirebaseEmailPassword$get_password()
```

Details: Get response from set_password

Returns: A list of length 2 containing `success` a boolean indicating whether setting password was successfully set and `response` containing `successful` as string or the error.

Method `re_authenticate()`**:**

Usage:

```
FirebaseEmailPassword$re_authenticate(password)
```

Arguments:

`password` The authenticated user password, the user should be prompted to enter it.

Details: Re-authenticate the user.
Some security-sensitive actions—such as deleting an account, setting a primary email address, and changing a password—require that the user has recently signed in. If you perform one of these actions, and the user signed in too long ago, the action fails with an error.

Returns: self

Method `get_re_authenticated()`**:**

Usage:

`FirebaseEmailPassword$get_re_authenticated()`

Details: Get response from re_authenticate

Returns: A list of length 2 containing `success` a boolean indicating whether re-authentication was successful and `response` containing `successful` as string or the error.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FirebaseEmailPassword$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

Also signs in the user if successful.

Examples

```
library(shiny)
library(firebase)

# modals
register <- modalDialog(
  title = "Register",
  textInput("email_create", "Your email"),
  passwordInput("password_create", "Your password"),
  footer = actionButton("create", "Register")
)

sign_in <- modalDialog(
  title = "Sign in",
  textInput("email_signin", "Your email"),
  passwordInput("password_signin", "Your password"),
  footer = actionButton("signin", "Sign in")
)

ui <- fluidPage(
  useFirebase(), # import dependencies
  actionButton("register_modal", "Register"),
  actionButton("signin_modal", "Signin"),
  plotOutput("plot")
)

server <- function(input, output){

  f <- FirebaseEmailPassword$new()

  # open modals
  observeEvent(input$register_modal, {
    showModal(register)
  })
```

```
  observeEvent(input$signin_modal, {
    showModal(sign_in)
  })

  # create the user
  observeEvent(input$create, {
    f$create(input$email_create, input$password_create)
  })

  # check if creation sucessful
  observeEvent(f$get_created(), {
    created <- f$get_created()

    if(created$success){
      removeModal()
      showNotification("Account created!", type = "message")
    } else {
      showNotification("Error!", type = "error")
    }

    # print results to the console
    print(created)
  })

  observeEvent(input$signin, {
    removeModal()
    f$sign_in(input$email_signin, input$password_signin)
  })

  output$plot <- renderPlot({
    f$req_sign_in()
    plot(cars)
  })

}

## Not run: shinyApp(ui, server)
```

`FirebaseOauthProviders`

OAuth Providers

Description

Use OAuth provides such as Github or Facebook to allow users to conveniently sign in.

Value

An object of class `FirebaseOauthProviders`.

Super classes

firebase::Firebase -> firebase::FirebaseAuth -> FirebaseOauthProviders

Methods

Public methods:

- FirebaseOauthProviders$new()
- FirebaseOauthProviders$set_provider()
- FirebaseOauthProviders$launch()
- FirebaseOauthProviders$clone()

Method new():

Usage:

```
FirebaseOauthProviders$new(
  persistence = c("session", "local", "memory"),
  config_path = "firebase.rds",
  language_code = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

persistence How the auth should persit: none, the user has to sign in at every visit, session will only persist in current tab, local persist even when window is closed.

config_path Path to the configuration file as created by firebase_config.

language_code Sets the language to use for the UI. Supported languages are listed here. Set to browser to use the default browser language of the user.

session A valid shiny session.

Details: Initialiases Firebase Email Link
Initialises the Firebase application client-side.

Method set_provider():

Usage:

```
FirebaseOauthProviders$set_provider(provider, ...)
```

Arguments:

provider The provider to user, e.g.: microsoft.com, yahoo.com or google.com.

... Additional options to pass to setCustomParameters.

Details: Define provider to use

Returns: self

Method launch():

Usage:

```
FirebaseOauthProviders$launch(
  flow = c("popup", "redirect"),
  get_credentials = FALSE
)
```

Arguments:

`flow` Authentication flow, either popup or redirect.

`get_credentials` Whether to extract underlying oauth credentials.

Details: Launch sign in with Google.

Returns: self

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FirebaseOauthProviders$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(shiny)
library(firebase)

ui <- fluidPage(
  useFirebase(),
  actionButton("signin", "Sign in with Microsoft", icon = icon("microsoft")),
  plotOutput("plot")
)

server <- function(input, output, session){
  f <- FirebaseOauthProviders$
    new()$
    set_provider("microsoft.com")

  observeEvent(input$signin, {
    f$launch()
  })

  output$plot <- renderPlot({
    f$req_sign_in()
    plot(cars)
  })

}

## Not run: shinyApp(ui, server)
```

`FirebasePhone` *Phone*

Description

Use mobile phone numbers to authenticate users.

Value

An object of class `FirebasePhone`.

Super classes

`firebase::Firebase` -> `firebase::FirebaseAuth` -> `FirebasePhone`

Methods

Public methods:

- `FirebasePhone$new()`
- `FirebasePhone$verify()`
- `FirebasePhone$get_recaptcha()`
- `FirebasePhone$confirm()`
- `FirebasePhone$get_verification()`
- `FirebasePhone$get_confirmation()`
- `FirebasePhone$clone()`

Method `new()`:

Usage:

```
FirebasePhone$new(
  persistence = c("session", "local", "memory"),
  config_path = "firebase.rds",
  language_code = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`persistence` How the auth should persit: `none`, the user has to sign in at every visit, `session` will only persist in current tab, `local` persist even when window is closed.

`config_path` Path to the configuration file as created by `firebase_config`.

`language_code` Sets the language to use for the UI. Supported languages are listed here. Set to `browser` to use the default browser language of the user.

`session` A valid shiny session.

Details: Initialiases Firebase Phone
Initialises the Firebase application client-side.

Method `verify()`:

Usage:

```
FirebasePhone$verify(number, id = NULL)
```

Arguments:

`number` Phone number of the user.

`id` Id of the button that triggers verification. If this is `NULL` the user has to go through the recaptcha, if not `NULL` is invisible.

Details: Verify a phhone number

Method `get_recaptcha()`**:**

Usage:

```
FirebasePhone$get_recaptcha()
```

Details: Results from the recaptcha

Method `confirm()`**:**

Usage:

```
FirebasePhone$confirm(code)
```

Arguments:

`code` Confirmation code received by the user.

Details: Confirm a code

Method `get_verification()`**:**

Usage:

```
FirebasePhone$get_verification()
```

Details: Get Verification

Returns: A list with a boolean (`success`) indicating whether the operation was successful and a `response` contianing the response from Firebase.

Method `get_confirmation()`**:**

Usage:

```
FirebasePhone$get_confirmation()
```

Details: Get Confirmation

Returns: A list with a boolean (`success`) indicating whether the operation was successful and a `response` contianing the response from Firebase.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FirebasePhone$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FirebaseSocial` *Social*

Description

Use social sites for authentication.

Value

An object of class `FirebaseSocial`.

Super classes

firebase::Firebase -> firebase::FirebaseAuth -> FirebaseSocial

Methods

Public methods:

- FirebaseSocial$new()
- FirebaseSocial$set_scope()
- FirebaseSocial$launch_google()
- FirebaseSocial$launch_github()
- FirebaseSocial$launch_facebook()
- FirebaseSocial$launch_twitter()
- FirebaseSocial$clone()

Method new():

Usage:

```
FirebaseSocial$new(
  persistence = c("session", "local", "memory"),
  config_path = "firebase.rds",
  language_code = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

persistence How the auth should persit: none, the user has to sign in at every visit, session will only persist in current tab, local persist even when window is closed.

config_path Path to the configuration file as created by firebase_config.

language_code Sets the language to use for the UI. Supported languages are listed here. Set to browser to use the default browser language of the user.

session A valid shiny session.

Details: Initialiases Firebase Social
Initialises the Firebase application client-side.

Method set_scope():

Usage:

```
FirebaseSocial$set_scope(scope)
```

Arguments:

scope Google scope.

Details: Define the scope to request from Google.

Returns: self

Method launch_google():

Usage:

```
FirebaseSocial$launch_google(flow = c("popup", "redirect"))
```

Arguments:

`flow` Authentication flow, either popup or redirect.

Details: Launch sign in with Google.

Returns: self

Method `launch_github()`**:**

Usage:

```
FirebaseSocial$launch_github(flow = c("popup", "redirect"))
```

Arguments:

`flow` Authentication flow, either popup or redirect.

Details: Launch sign in with Github.

Returns: self

Method `launch_facebook()`**:**

Usage:

```
FirebaseSocial$launch_facebook(flow = c("popup", "redirect"))
```

Arguments:

`flow` Authentication flow, either popup or redirect.

Details: Launch sign in with Facebook.

Returns: self

Method `launch_twitter()`**:**

Usage:

```
FirebaseSocial$launch_twitter(flow = c("popup", "redirect"))
```

Arguments:

`flow` Authentication flow, either popup or redirect.

Details: Launch sign in with Facebook.

Returns: self

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FirebaseSocial$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(shiny)
library(firebase)

# define signin
signin <- modalDialog(
  title = "Login",
  actionButton("google", "Google", icon = icon("google"), class = "btn-danger"),
  actionButton("github", "Github", icon = icon("github")),
  footer = NULL
)

ui <- fluidPage(
  useFirebase()
)

server <- function(input, output) {
  showModal(signin)

  f <- FirebaseSocial$new()

  observeEvent(input$google, {
    f$launch_google()
  })

  observeEvent(input$github, {
    f$launch_github()
  })
}

## Not run: shinyApp(ui, server)
```

`FirebaseUI` *Prebuilt UI*

Description

Use firebase to manage authentications.

Value

An object of class `FirebaseUI`.

Super classes

`firebase::Firebase` -> `firebase::FirebaseAuth` -> `FirebaseUI`

Public fields

`tos_url` URL to the Terms of Service page.

`privacy_policy_url` The URL to the Privacy Policy page.

Methods

Public methods:

- `FirebaseUI$new()`
- `FirebaseUI$set_providers()`
- `FirebaseUI$set_tos_url()`
- `FirebaseUI$set_privacy_policy_url()`
- `FirebaseUI$launch()`
- `FirebaseUI$reset_password()`
- `FirebaseUI$get_reset()`
- `FirebaseUI$send_verification_email()`
- `FirebaseUI$get_verification_email()`
- `FirebaseUI$set_password()`
- `FirebaseUI$get_password()`
- `FirebaseUI$re_authenticate()`
- `FirebaseUI$get_re_authenticated()`
- `FirebaseUI$clone()`

Method `new()`:

Usage:

```
FirebaseUI$new(
  persistence = c("session", "local", "memory"),
  config_path = "firebase.rds",
  language_code = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`persistence` How the auth should persit: `none`, the user has to sign in at every visit, `session` will only persist in current tab, `local` persist even when window is closed.

`config_path` Path to the configuration file as created by `firebase_config`.

`language_code` Sets the language to use for the UI. Supported languages are listed here. Set to `browser` to use the default browser language of the user.

`session` A valid shiny session.

Details: Initialiases Firebase UI
Initialises the Firebase application client-side.

Method `set_providers()`:

Usage:

```
FirebaseUI$set_providers(
  google = FALSE,
  facebook = FALSE,
  twitter = FALSE,
  github = FALSE,
  email = FALSE,
  email_link = FALSE,
  microsoft = FALSE,
  apple = FALSE,
  yahoo = FALSE,
  phone = FALSE,
  anonymous = FALSE
)
```

Arguments:

`google`, `facebook`, `twitter`, `github`, `email`, `email_link`, `microsoft`, `apple`, `yahoo`, `phone`, `anonymous` Set to `TRUE` the providers you want to use, at least one.

Details: Define signin and login providers.

Returns: self

Method `set_tos_url()`**:**

Usage:

```
FirebaseUI$set_tos_url(url)
```

Arguments:

`url` URL to use.

Details: Defines Tterms of Services URL

Returns: self

Method `set_privacy_policy_url()`**:**

Usage:

```
FirebaseUI$set_privacy_policy_url(url)
```

Arguments:

`url` URL to use.

Details: Defines Privacy Policy URL

Returns: self

Method `launch()`**:**

Usage:

```
FirebaseUI$launch(flow = c("popup", "redirect"), account_helper = FALSE)
```

Arguments:

`flow` The signin flow to use, popup or redirect.

`account_helper` Wether to use accountchooser.com upon signing in or signing up with email, the user will be redirected to the accountchooser.com website and will be able to select one of their saved accounts. You can disable it by specifying the value below.

... Any other option to pass to Firebase UI.

Details: Setup the signin form.

Returns: self

Method `reset_password()`**:**

Usage:

```
FirebaseUI$reset_password(email = NULL)
```

Arguments:

`email` Email to send reset link to, if missing looks for current logged in user's email

Details: Reset user password

Returns: self

Method `get_reset()`**:**

Usage:

```
FirebaseUI$get_reset()
```

Details: Get whether password reset email was successfully sent

Returns: A list of length 2 containing `success` a boolean indicating whether email reset was successful and `response` containing `successful` or the error.

Method `send_verification_email()`**:**

Usage:

```
FirebaseUI$send_verification_email()
```

Details: Send the user a verification email

Returns: self

Method `get_verification_email()`**:**

Usage:

```
FirebaseUI$get_verification_email()
```

Details: Get result of verification email sending procedure

Returns: A list of length 2 containing `success` a boolean indicating whether email verification was successfully sent and `response` containing `successful` or the error.

Method `set_password()`**:**

Usage:

```
FirebaseUI$set_password(password)
```

Arguments:

`password` The authenticated user password, the user should be prompted to enter it.

Details: Set user password
Useful to provide ability to change password.

Returns: self

Method `get_password()`**:**

Usage:

```
FirebaseUI$get_password()
```

Details: Get response from set_password

Returns: A list of length 2 containing `success` a boolean indicating whether setting password was successfully set and `response` containing `successful` as string or the error.

Method `re_authenticate()`**:**

Usage:

```
FirebaseUI$re_authenticate(password)
```

Arguments:

`password` The authenticated user password, the user should be prompted to enter it.

Details: Re-authenticate the user.
Some security-sensitive actions—such as deleting an account, setting a primary email address, and changing a password—require that the user has recently signed in. If you perform one of these actions, and the user signed in too long ago, the action fails with an error.

Method `get_re_authenticated()`**:**

Usage:

```
FirebaseUI$get_re_authenticated()
```

Details: Get response from re_authenticate

Returns: A list of length 2 containing `success` a boolean indicating whether re-authentication was successful and `response` containing `successful` as string or the error.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FirebaseUI$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(shiny)
library(firebase)

ui <- fluidPage(
  useFirebase(), # import dependencies
  firebaseUIContainer() # import UI
)

server <- function(input, output){
  f <- FirebaseUI$
    new()$ # instantiate
    set_providers( # define providers
      email = TRUE,
      google = TRUE
    )
```

```
}

## Not run: shinyApp(ui, server)
```

`RealtimeDatabase` *Realtime Database*

Description

Access Firebase Realtime Database

Value

An object of class `RealtimeDatabase`.

Super class

`firebase::Firebase`

Methods

Public methods:

- `RealtimeDatabase$new()`
- `RealtimeDatabase$ref()`
- `RealtimeDatabase$on_value()`
- `RealtimeDatabase$set()`
- `RealtimeDatabase$update()`
- `RealtimeDatabase$delete()`
- `RealtimeDatabase$clone()`

Method `new()`:

Usage:

```
RealtimeDatabase$new(
  config_path = "firebase.rds",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`config_path` Path to the configuration file as created by `firebase_config`.

`session` A valid shiny session.

Details: Initialiases Firebase Storage
Initialises the Firebase Storage application client-side.

Method `ref()`:

Usage:

```
RealtimeDatabase$ref(path = NULL)
```

Arguments:

`path` Path to the database full URL to file.

Details: Reference
Creates a reference to a file or directory you want to operate on. Note that this reference persists, make sure you change it between operations.

Returns: Invisibly return the class instance.

Method `on_value()`**:**

Usage:

```
RealtimeDatabase$on_value(response, path = NULL)
```

Arguments:

`response` A boolean or character string. `TRUE` indicates that you want to capture the results of the file upload (e.g.: success or failed) with `get_response` method. `FALSE` indicates you do not want those results back. A character string is used as named of the response which then can be used in the `get_response` method.

`path` Path to the database full URL to file.

Details: On Value
When `path` or `ref` sees an update it sends the new data to `response`.

Method `set()`**:**

Usage:

```
RealtimeDatabase$set(data, response = NULL, path = NULL)
```

Arguments:

`data` Dataset to upload.

`response` A boolean or character string. `TRUE` indicates that you want to capture the results of the file upload (e.g.: success or failed) with `get_response` method. `FALSE` indicates you do not want those results back. A character string is used as named of the response which then can be used in the `get_response` method.

`path` Path to the database full URL to file.

Details: Set Data
Pushes data to the database.

Method `update()`**:**

Usage:

```
RealtimeDatabase$update(data, response = NULL, path = NULL)
```

Arguments:

`data` Dataset to update.

`response` A boolean or character string. `TRUE` indicates that you want to capture the results of the file upload (e.g.: success or failed) with `get_response` method. `FALSE` indicates you do not want those results back. A character string is used as named of the response which then can be used in the `get_response` method.

`path` Path to the database full URL to file.

Details: Update Data
Update a record.

Method `delete()`**:**

Usage:

```
RealtimeDatabase$delete(response = NULL, path = NULL)
```

Arguments:

`response` A boolean or character string. `TRUE` indicates that you want to capture the results of the file upload (e.g.: success or failed) with `get_response` method. `FALSE` indicates you do not want those results back. A character string is used as named of the response which then can be used in the `get_response` method.

`path` Path to the database full URL to file.

Details: Delete
Delete data to the database.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RealtimeDatabase$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`recaptcha` *Recaptcha*

Description

Add the recaptcha, require for the `FirebasePhone` class.

Usage

```
recaptchaUI(
  ns = function(x) {
     x
 }
)
```

Arguments

`ns`	Namespace, optional, only required if using this function in multiple places.

reqSignin *Requires Signin*

Description

Define UI element that require the user to be signed in. This will hide them *viusally* until the user signs in. Note that this is not secure as someone can easily change the CSS when visiting the page to reveal those elements.

Usage

```
reqSignin(...)
```

Arguments

...	Any valid tags.

Value

No return value, called for side effects.

See Also

reqSignout

reqSignout *Requires Signout*

Description

Define UI element that requires *no* user to be signed in. This will hide them *viusally* if no user is signed in. Note that this is not secure as someone can easily change the CSS when visiting the page to reveal those elements.

Usage

```
reqSignout(...)
```

Arguments

...	Any valid tags.

See Also

reqSignin

Storage *Storage*

Description

Storage

Storage

Value

An object of class `Storage`.

Super class

`firebase::Firebase` -> `Storage`

Methods

Public methods:

- `Storage$new()`
- `Storage$ref()`
- `Storage$upload_file()`
- `Storage$download_file()`
- `Storage$delete_file()`
- `Storage$get_metadata()`
- `Storage$list_files_all()`
- `Storage$get_response()`
- `Storage$clone()`

Method `new()`:

Usage:

```
Storage$new(
  config_path = "firebase.rds",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`config_path` Path to the configuration file as created by `firebase_config`.

`session` A valid shiny session.

Details: Initialiases Firebase Storage
Initialises the Firebase Storage application client-side.

Method `ref()`:

Usage:

```
Storage$ref(path = NULL)
```

Arguments:

`path` Path to the file, directory, bucket, or full URL to file. If `NULL` creates a path to the root.

Details: Reference
Creates a reference to a file or directory you want to operate on. Note that this reference persists, make sure you change it between operations.

Returns: Invisibly return the class instance.

Method `upload_file()`**:**

Usage:

```
Storage$upload_file(file, response = TRUE)
```

Arguments:

`file` Path to the file to upload.

`response` A boolean or character string. `TRUE` indicates that you want to capture the results of the file upload (e.g.: success or failed) with `get_response` method. `FALSE` indicates you do not want those results back. A character string is used as named of the response which then can be used in the `get_response` method.

Details: Upload a File
Upload a file to the store system or bucket. Requires a valid authentication.

Examples:

```
\dontrun{
s <- Storage$new()

# default response
s$
  ref("test.png")$
  upload_file("path/to/file.png")

observeEvent(s$get_response(), {
  # do something
})

# named response
s$
  ref("test.png")$
  upload_file("path/to/file.png", response = "fl")

observeEvent(s$get_response("fl"), {
  # do something
})
}
```

Method `download_file()`**:**

Usage:

```
Storage$download_file(response = TRUE)
```

Arguments:

`response` A boolean or character string. `TRUE` indicates that you want to capture the results of the file upload (e.g.: success or failed) with `get_response` method. `FALSE` indicates you do not want those results back. A character string is used as named of the response which then can be used in the `get_response` method.

Details: Download a File
Download a file from the store system or bucket. Requires a valid authentication.

Examples:

```
\dontrun{
s <- Storage$new()

s$
  ref("test.png")$
  upload_file("path/to/file.png")$
  download_file("dl")

observeEvent(s$get_response("dl"), {
  # do something
})
}
```

Method `delete_file()`**:**

Usage:

```
Storage$delete_file(response = TRUE)
```

Arguments:

`response` A boolean or character string. `TRUE` indicates that you want to capture the results of the file upload (e.g.: success or failed) with `get_response` method. `FALSE` indicates you do not want those results back. A character string is used as named of the response which then can be used in the `get_response` method.

Details: Delete a File
Delete a file from the store system or bucket. Requires a valid authentication.

Method `get_metadata()`**:**

Usage:

```
Storage$get_metadata(response = TRUE)
```

Arguments:

`response` A boolean or character string. `TRUE` indicates that you want to capture the results of the file upload (e.g.: success or failed) with `get_response` method. `FALSE` indicates you do not want those results back. A character string is used as named of the response which then can be used in the `get_response` method.

Details: File Metadata
Get the metadata of a file from the store system or bucket. Requires a valid authentication.

Method `list_files_all()`**:**

Usage:

```
Storage$list_files_all(response = TRUE)
```

Arguments:

`response` A boolean or character string. `TRUE` indicates that you want to capture the results of the file upload (e.g.: success or failed) with `get_response` method. A character string is used as named of the response which then can be used in the `get_response` method.

Details: List All Files
List all files in the reference (`ref`). Requires a valid authentication.

Method `get_response()`**:**

Usage:

```
Storage$get_response(response = NULL)
```

Arguments:

`response` Name of the response.

Details: Capture response

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Storage$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `Storage$upload_file`
## ------------------------------------------------

## Not run:
s <- Storage$new()

# default response
s$
  ref("test.png")$
  upload_file("path/to/file.png")

observeEvent(s$get_response(), {
  # do something
})

# named response
s$
  ref("test.png")$
  upload_file("path/to/file.png", response = "fl")

observeEvent(s$get_response("fl"), {
  # do something
})

## End(Not run)
```

```
## ------------------------------------------------
## Method `Storage$download_file`
## ------------------------------------------------

## Not run:
s <- Storage$new()

s$
  ref("test.png")$
  upload_file("path/to/file.png")$
  download_file("dl")

observeEvent(s$get_response("dl"), {
  # do something
})

## End(Not run)
```

Index