

Package ‘firebase’

October 24, 2021

Title Integrates 'Google Firebase' Authentication Method with 'Shiny'

Version 0.2.1

Date 2021-10-24

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'.

License AGPL-3

Encoding UTF-8

Imports R6, cli, shiny, openssl, jose

RoxygenNote 7.1.2

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

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

NeedsCompilation no

Author John Coene [aut, cre]

Maintainer John Coene <jcoenep@gmail.com>

Repository CRAN

Date/Publication 2021-10-24 19:00:02 UTC

R topics documented:

check_urls	2
config	2
dependencies	3
Firebase	4
FirebaseEmailLink	7
FirebaseEmailPassword	10
FirebaseOAuthProviders	15
FirebaseSocial	16
FirebaseUI	18

recaptcha	22
reqSignIn	23
reqSignout	23

Index	24
--------------	-----------

check_urls	<i>Check URLs</i>
------------	-------------------

Description

Check that tos and privacy policy urls are set.

Usage

```
check_urls(firebase_ui)
```

Arguments

firebase_ui An object of class [FirebaseUI](#).

config	<i>Config</i>
--------	---------------

Description

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

Usage

```
create_config(api_key, project_id, auth_domain = NULL, overwrite = FALSE)
```

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

Arguments

api_key	API key of your project.
project_id	Id of your web project.
auth_domain	Authentication domain, if omitted uses, attempts to build firebase's default domain.
overwrite	Whether to overwrite any existing configuration file.

Details

Do not share this file with anyone.

Value

Path to file.

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(container = TRUE)
```

```
firebaseUIContainer()
```

Arguments

`analytics` Whether to include analytics.

`firestore` Whether to include firestore.

`container` Whether to include the container that wraps the pre-built UI. If set to FALSE then one must use `firebaseUIContainer` where said pre-built UI is required.

Details

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

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

Use firebase to manage authentications.

Public fields

session A valid Shiny session.

Active bindings

signed_in Read the signed in user.

signed_up Read the signed in user.

Methods

Public methods:

- `Firebase$new()`
- `Firebase$print()`
- `Firebase$sign_out()`
- `Firebase$get_sign_out()`
- `Firebase$get_signed_in()`
- `Firebase$get_signed_up()`
- `Firebase$is_signed_in()`
- `Firebase$req_sign_in()`
- `Firebase$req_sign_out()`
- `Firebase$set_language_code()`
- `Firebase$delete_user()`
- `Firebase$get_delete_user()`
- `Firebase$get_access_token()`
- `Firebase$clear()`
- `Firebase$request_id_token()`
- `Firebase$get_id_token()`
- `Firebase$clone()`

Method `new()`:

Usage:

```
Firebase$new(  
  persistence = c("none", "session", "local"),  
  config_path = "firebase.rds",  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments:

persistence How the auth should persist: 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](#).

session A valid shiny session.

Details: Initialise firebase

Method print():

Usage:

```
Firestore#print()
```

Details: Print the class

Method sign_out():

Usage:

```
Firestore#sign_out()
```

Details: Signs out user

Returns: self

Method get_sign_out():

Usage:

```
Firestore#get_sign_out()
```

Details: Get signed out results

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

Method get_signed_in():

Usage:

```
Firestore#get_signed_in()
```

Details: Signed in user details triggered when auth states changes

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

Method get_signed_up():

Usage:

```
Firestore#get_signed_up()
```

Details: Get results of a sign up

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

Method is_signed_in():

Usage:

```
Firestore$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:

```
Firestore$req_sign_in()
```

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

Method req_sign_out():

Usage:

```
Firestore$req_sign_out()
```

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

Method set_language_code():

Usage:

```
Firestore$set_language_code(code)
```

Arguments:

code iso639-1 language code.

Details: Set language code for auth provider

Returns: self

Method delete_user():

Usage:

```
Firestore$delete_user()
```

Details: Delete the user

Returns: self

Method get_delete_user():

Usage:

```
Firestore$get_delete_user()
```

Details: Get result of user deletion

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

Method get_access_token():

Usage:

```
Firestore$get_access_token()
```

Details: Get user access token

Returns: User's access token

Method clear():

Usage:

Firestore\$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:

Firestore\$request_id_token()

Details: Request the users' ID Token

Used to retrieve 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:

Firestore\$get_id_token()

Details: Retrieve the users' ID Token

Also see request_id_token.

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

Usage:

Firestore\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

FirestoreEmailLink

Email Link

Description

Sign in the user by emailing them a link.

Super class

`firebase::Firestore` -> FirestoreEmailLink

Active bindings

email_verification Email verification results

email_sent Email send results

Methods

Public methods:

- `FirebaseEmailLink$config()`
- `FirebaseEmailLink$send()`
- `FirebaseEmailLink$get_email_sent()`
- `FirebaseEmailLink$get_email_verification()`
- `FirebaseEmailLink$clone()`

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()`:

Usage:

```
FirebaseEmailLink$send(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 whether 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 whether 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)

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())
  })
}

## Not run: shinyApp(ui, server)

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

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

## End(Not run)

## -----
## Method `FirebaseEmailLink$send`
## -----

## 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.

Super class

`firebase::Firebase` -> `FirebaseEmailPassword`

Active bindings

created Results of account creation

Methods**Public methods:**

- [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 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 whether 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 successful
  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)

```

FirestoreProviders
OAuth Providers

Description

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

Super class

`firebase::Firestore` -> FirestoreProviders

Methods

Public methods:

- `FirestoreProviders$set_provider()`
- `FirestoreProviders$launch()`
- `FirestoreProviders$clone()`

Method `set_provider()`:

Usage:

`FirestoreProviders$set_provider(provider)`

Arguments:

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

Details: Define provider to use

Returns: self

Method `launch()`:

Usage:

`FirestoreProviders$launch(flow = c("popup", "redirect"))`

Arguments:

flow Authentication flow, either popup or redirect.

Details: Launch sign in with Google.

Returns: self

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

Usage:

`FirestoreProviders$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)
```

FirebaseSocial

Social

Description

Use social sites for authentication.

Super class

`firebase::Firebase` -> `FirebaseSocial`

Methods

Public methods:

- `FirebaseSocial$set_scope()`
- `FirebaseSocial$launch_google()`
- `FirebaseSocial$launch_github()`
- `FirebaseSocial$launch_facebook()`
- `FirebaseSocial$launch_twitter()`

- [FirebaseSocial\\$clone\(\)](#)

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.

Super class

`firebase::Firebase` -> `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$set_providers()`
- `FirebaseUI$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 `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 Terms 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 Whether 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:

`FirestoreUI.send_verification_email()`

Details: Send the user a verification email

Returns: self

Method `get_verification_email():`

Usage:

`FirestoreUI.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:

`FirestoreUI.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:

`FirestoreUI.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:

`FirestoreUI.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
  useFirebaseUI() # import UI
)

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

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

recaptcha

Recaptcha

Description

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

Usage

```
recaptcha()
```

reqSignin	<i>Requires Signin</i>
-----------	------------------------

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](#).

See Also

[reqSignout](#)

reqSignout	<i>Requires Signout</i>
------------	-------------------------

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](#)

Index

`check_urls`, 2
`config`, 2
`create_config (config)`, 2

dependencies, 3

Firestore, 4
`firebase::Firestore`, 7, 10, 15, 16, 19
`firebase_config`, 5
`firebase_config (config)`, 2
`FirestoreEmailLink`, 7
`FirestoreEmailPassword`, 10
`FirestoreOauthProviders`, 15
`FirestoreSocial`, 16
`FirestoreUI`, 2, 3, 18
`firebaseUIContainer (dependencies)`, 3

`recaptcha`, 22
`reqSignIn`, 23, 23
`reqSignout`, 23, 23

tags, 23

`useFirestore (dependencies)`, 3
`useFirestoreUI (dependencies)`, 3