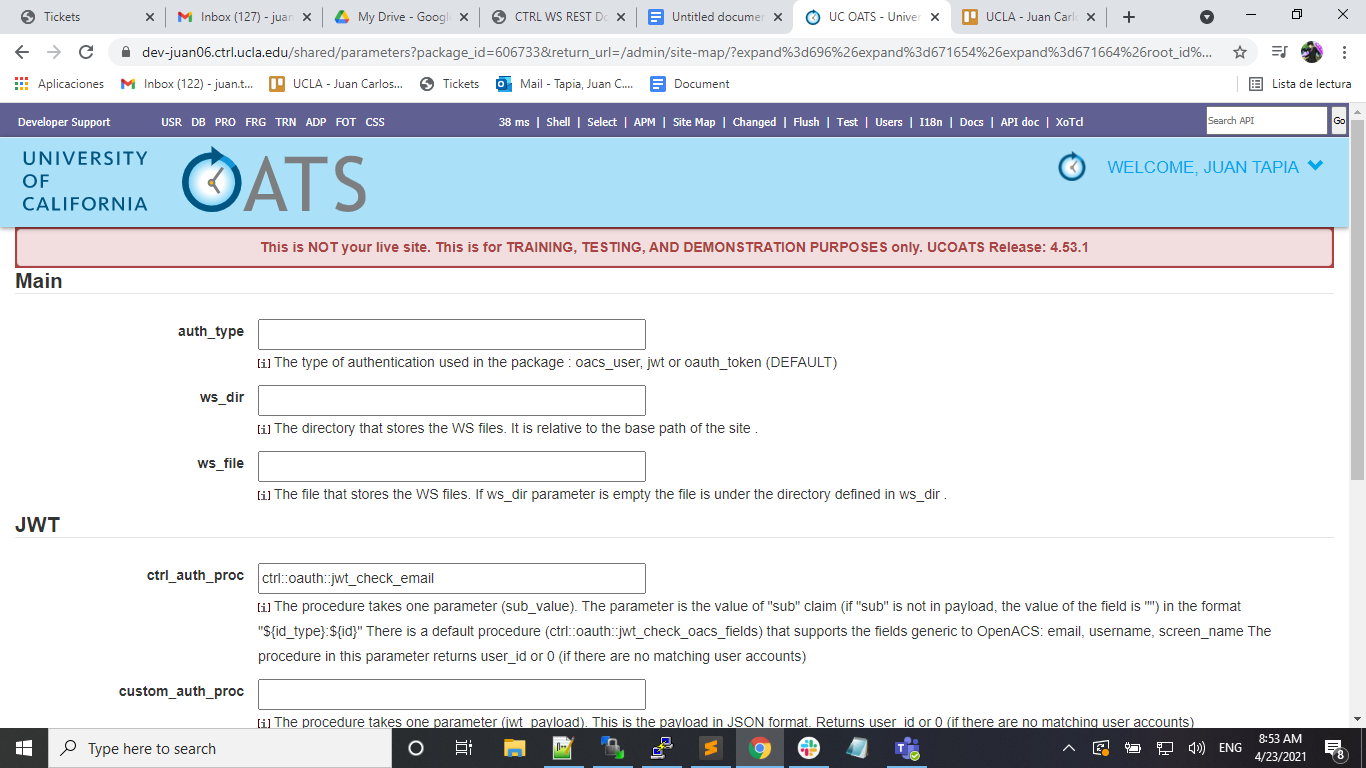
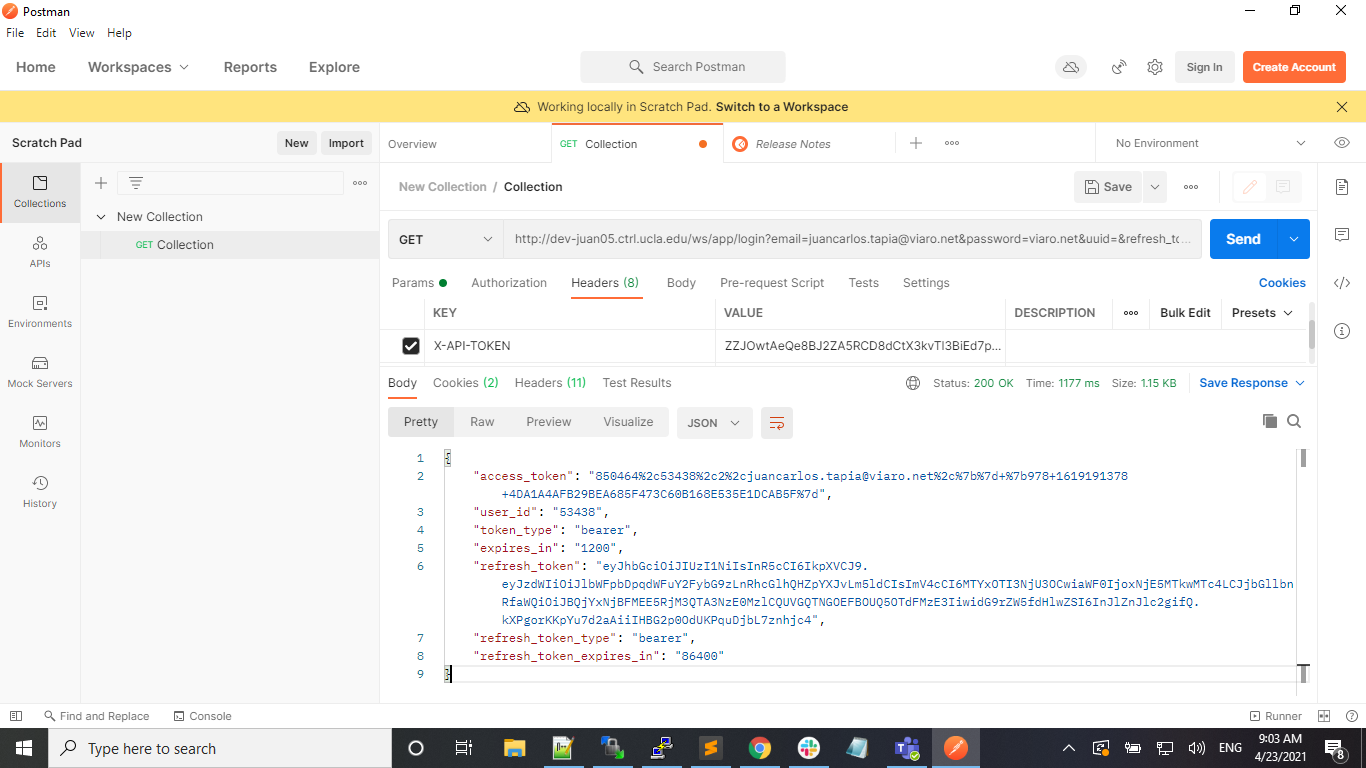
How to setup tokens

There are 3 options for token use in the ctrl-ws-restful package that can be changed in the package parameters.

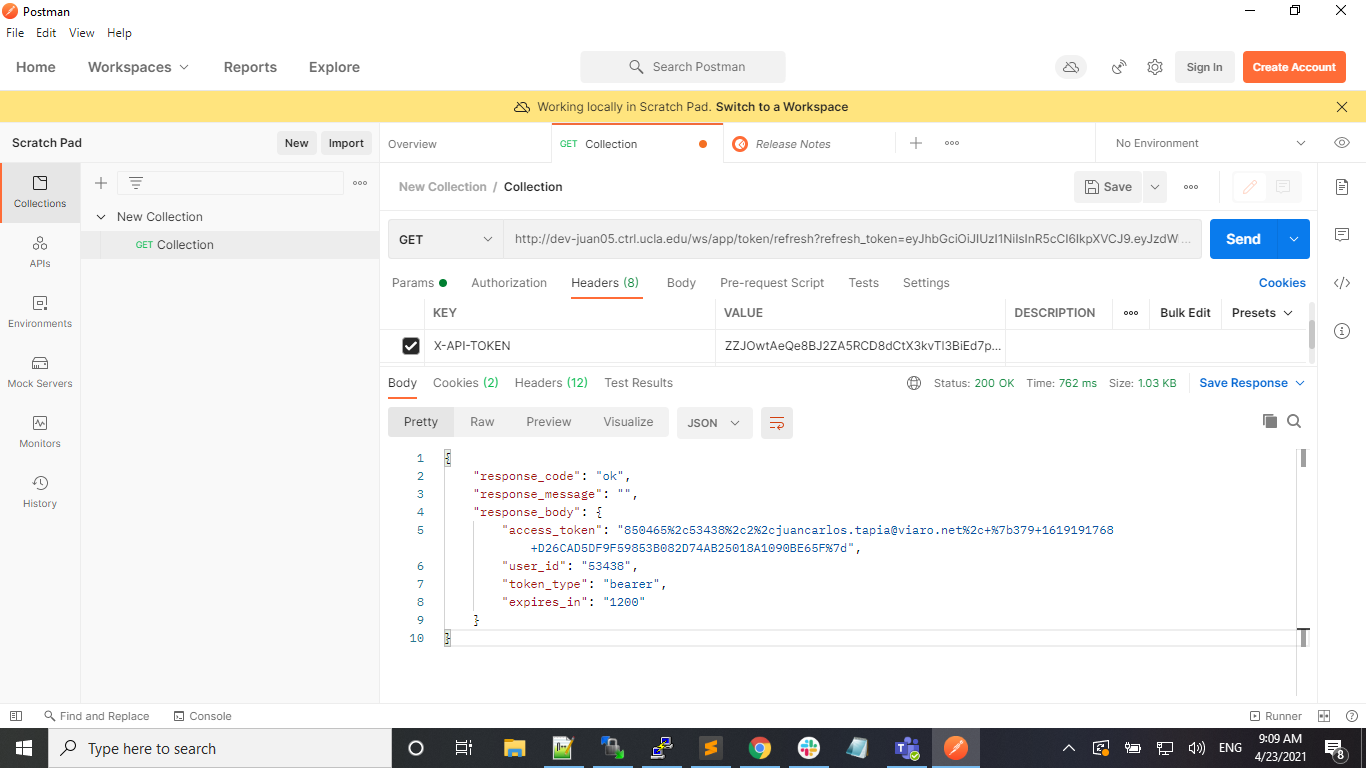


**oacs\_user**

* To create an authorization token the user needs to create an API endpoint that points to the procedure “ctrl::oauth::authenicate\_with\_token” that takes an “email, a “password” and an optional parameter “refresh\_token\_p” to indicate if we want an extra token for refreshing.
* If the email and the password are valid in the system, then the endpoint returns a token created using a hash of the user and session information.
* If the refresh\_token\_p parameter is 1, then the procedure also returns a JWT token that can be used for refreshing.

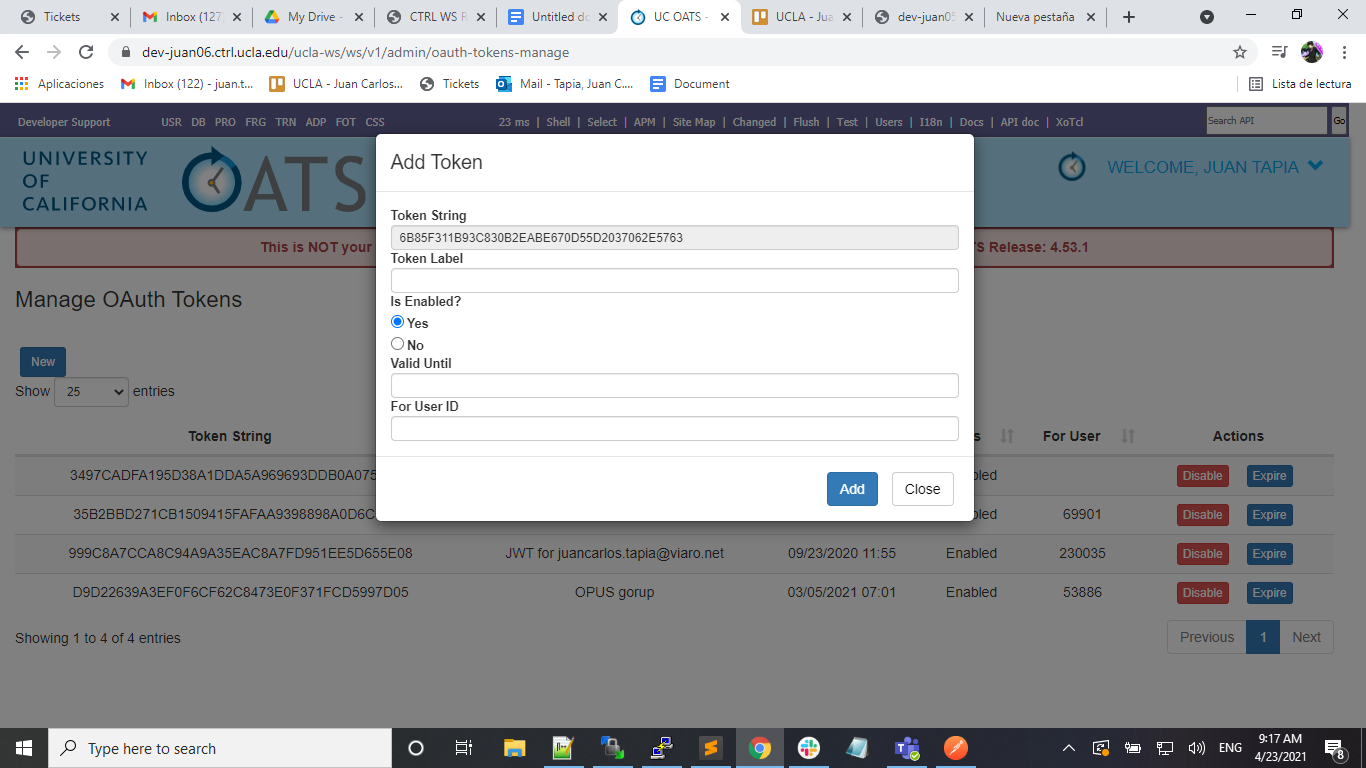


* The access\_token expires after 20 minutes (1200 seconds). The refresh\_token expires after 24 hours (86480 seconds).
* To use the refresh\_token, it is necessary to create another endpoint that points to the procedure “ctrl::oauth::refresh\_token” that takes refresh\_token as the parameter. If the refresh token is still valid then the procedure returns a new access\_token that again will expire in 20 minutes.

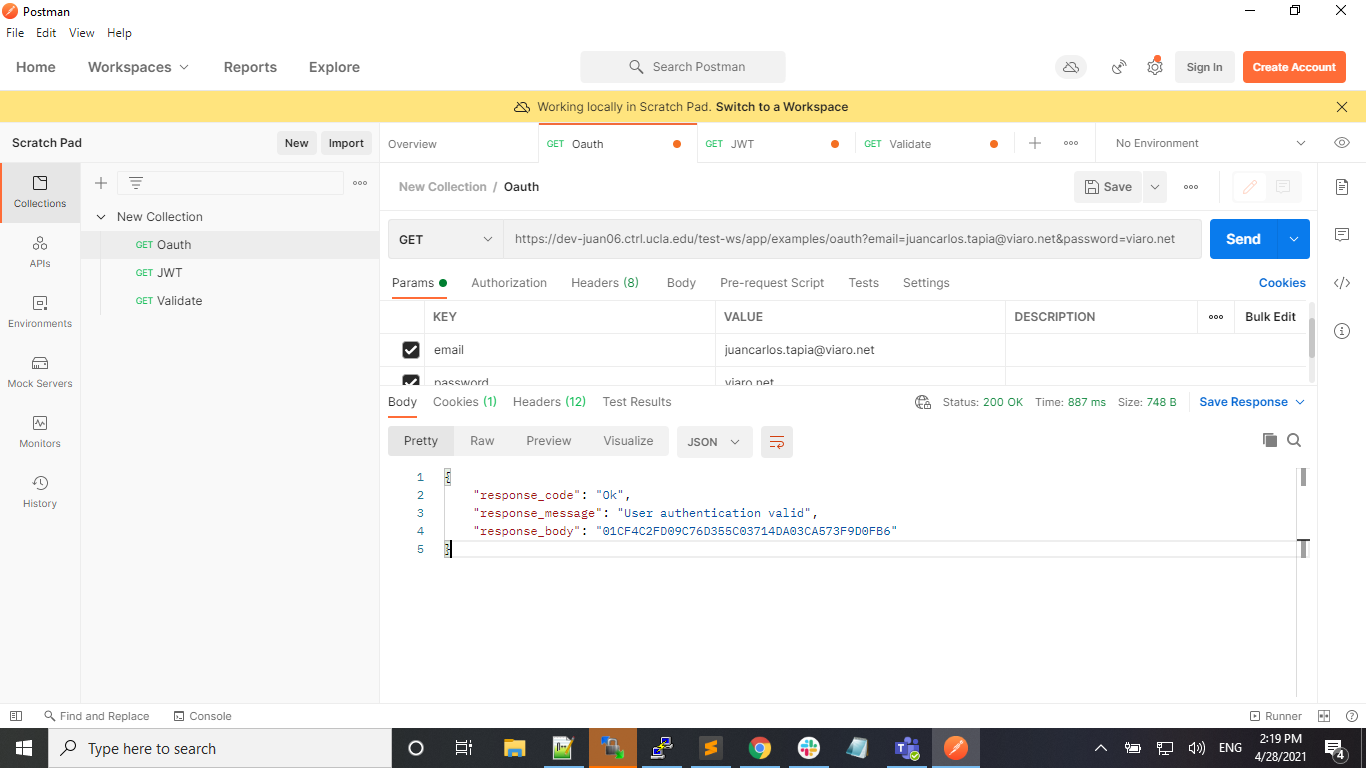


**oauth\_token**

* Go to the page /admin/oauth-tokens-manage in the package.
* When clicking the button New a popup window will appear and generate the authentication token automatically.
* In this popup we can set up a label for the token, an expiration date and assign it to a user\_id.



* To automatically create tokens then we need to create an endpoint to a procedure that creates Oauth tokens and returns them. There is a default procedure already “ctrl::oauth::login” that takes an email and a password and returns a token if the credentials passed are valid. On package installation there is a test endpoint named “app/examples/oauth”.

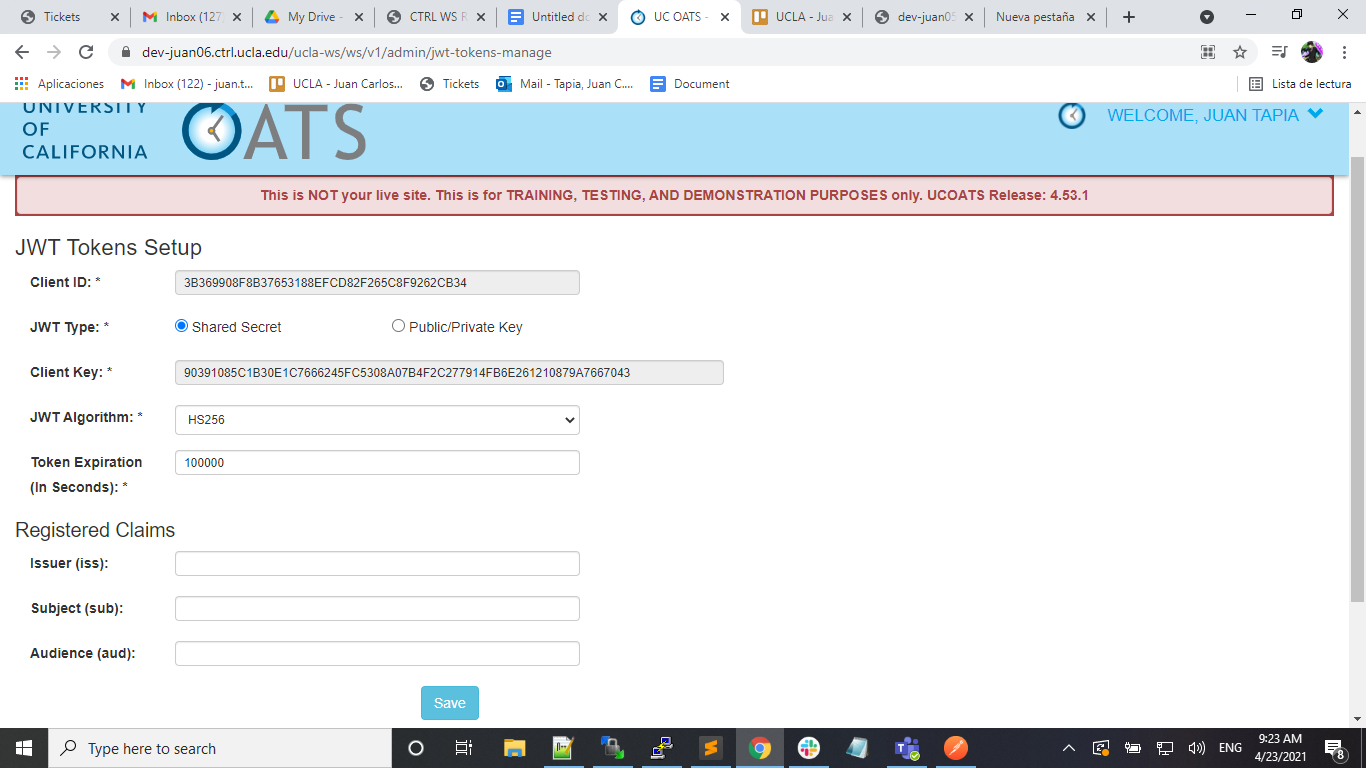


* Another way to create a token is to go to /admin/generate-token. The page will automatically create a token for the current user logged in.

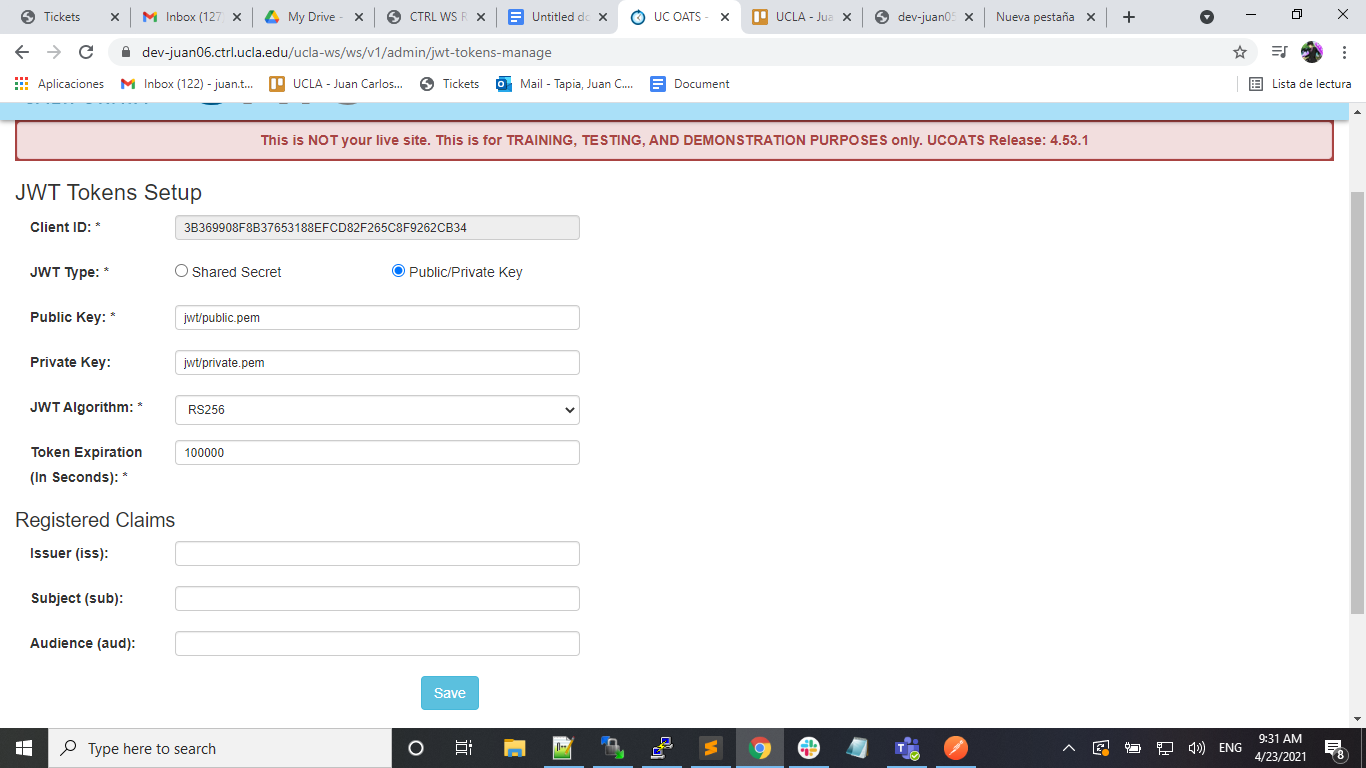
**jwt**

* Go to the page /admin/jwt-tokens-manage in the package.
* In this page we can set how the tokens will be created.
* The Client ID is a random value that will be unique to this package instance.
* The JWT Type decides how the tokens will be generated and validated.
  + If Shared Secret is selected then the system creates a random key that will be used to encode the token and will also be used to validate it.
  + If Public/Private Key is selected then we need to input the paths to 2 files that are relative to the project directory. The files are the Private Key to encode the tokens and the Public Key to decode the tokens.
* The JWT Algorithm indicates what algorithm will be used to encode and decode the tokens. The algorithm used will be stored in the token.
* The Token Expiration(In Seconds) indicates for how long the token will be valid.
* The fields Issuer (iss), Subject (sub) and Audience (aud) are optional fields that can be added to the to store more information in the Token Payload. They are static, but there is a way to automatically set the subject when tokens are generated automatically.

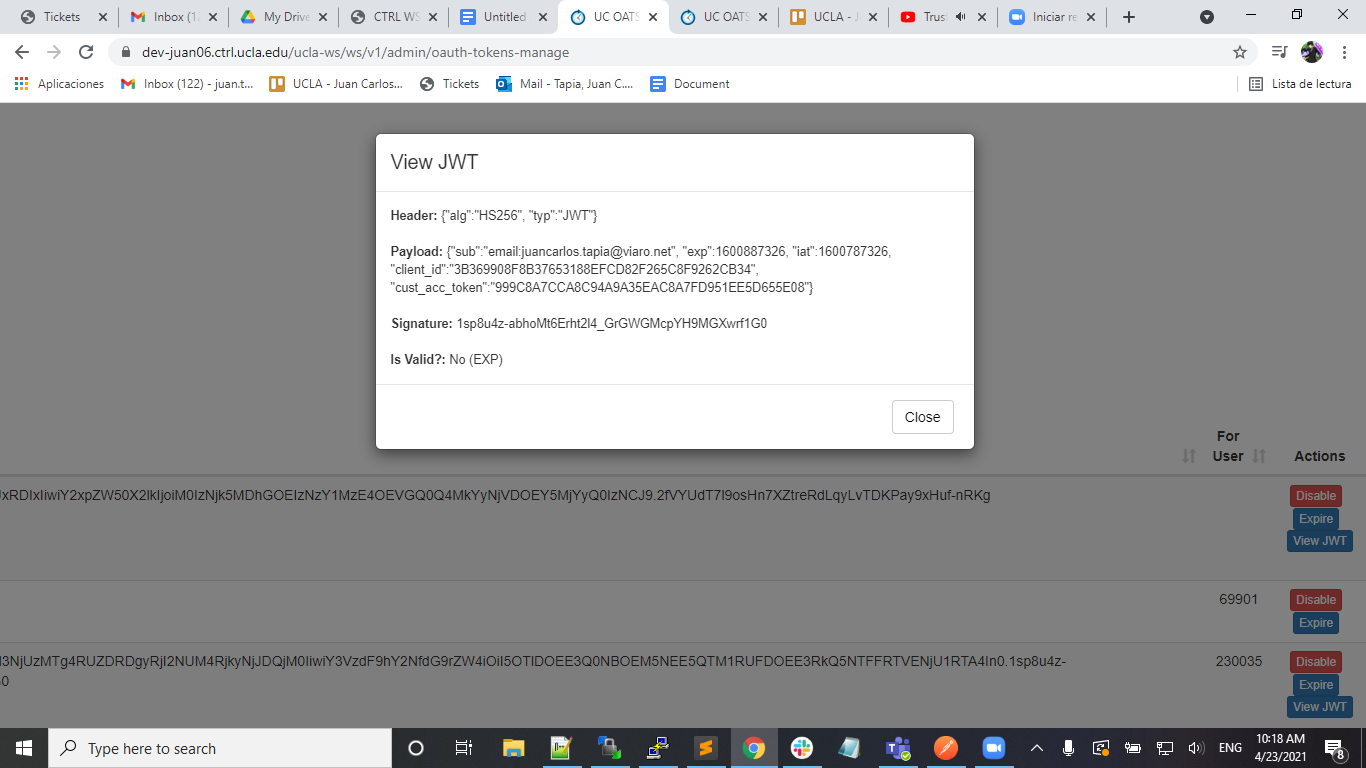
Shared Secret UI



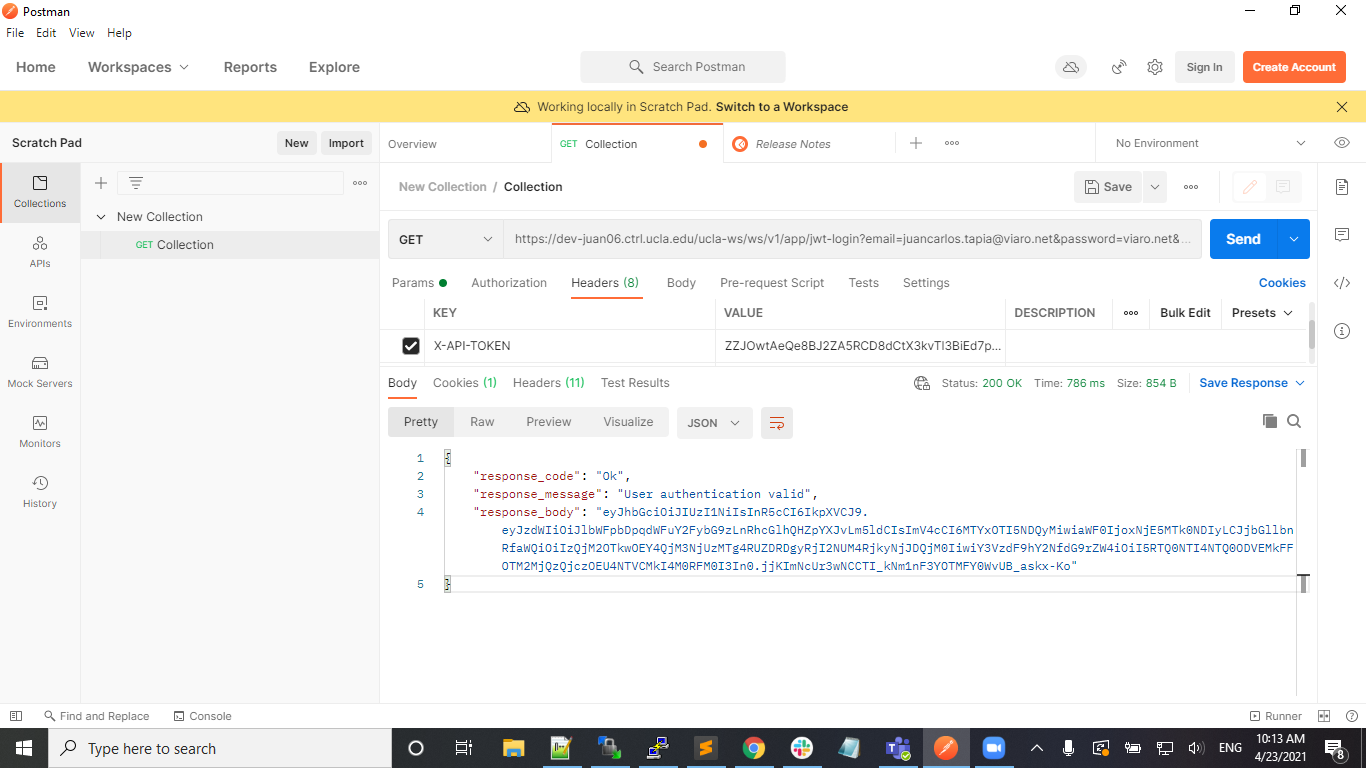
Public/Private Key UI



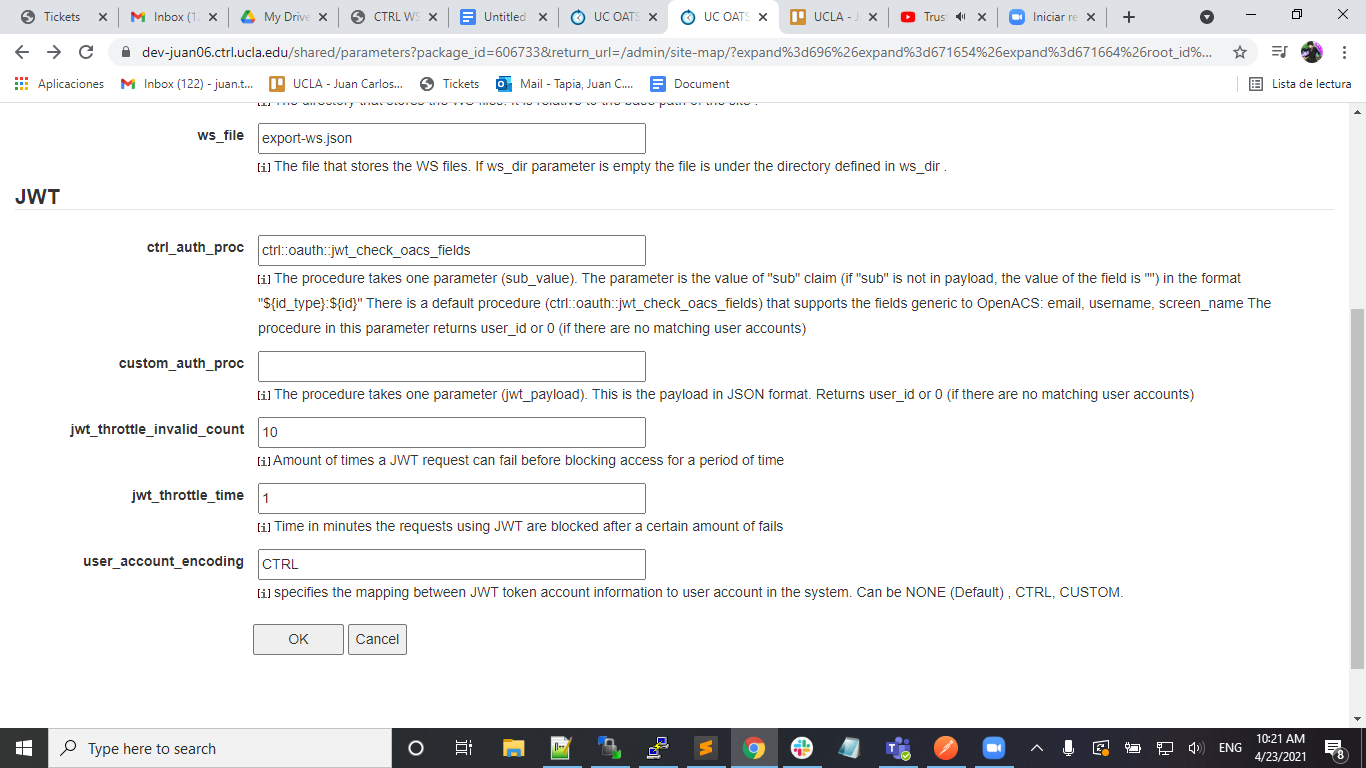
* To manually create a token go to the page /admin/oauth-tokens-manage and use the same process to create Oauth Tokens. These tokens can be used as both Oauth Tokens and JWT Tokens.
* There is a new field in the popup to select which user field to store in the token (email or username). The user information is stored in the “sub” claim in the token payload as “"sub":"email:[test@test.net](mailto:test@test.net)"” or “"sub":"username:[test@test.net](mailto:test@test.net)"”
* To view a JWT information and if it is still valid, in that same page click on the button “View JWT” and that will open a popup with the token information decoded.



* To automatically create tokens then we need to create an endpoint to a procedure that creates JWT tokens and returns them. There is a default procedure already “ctrl::restful::jwt::login\_default\_type\_email” that takes an email and a password and returns a token if the credentials passed are valid. On package installation there is a test endpoint named “app/examples/jwt”.



* Another way to create a token is to go to /admin/generate-token. The page will automatically create a token for the current user logged in.
* There are parameters in the ctrl-ws-package that are used in the validation of JWT.



* user\_acount\_encoding indicates how the system will check the user information in the token.
  + If NONE, then we don’t look for user information in the token.
  + If CTRL, then we used the procedure set in ctrl\_auth\_proc.
  + If CUSTOM, then we used the procedure defined in custom\_auth\_proc.
* ctrl\_auth\_proc is a procedure that must expect the parameter sub\_value. When validating the token, the system will pass to it the value in the “sub” claim. There is a default procedure “ctrl::oauth::jwt\_check\_oacs\_fields" that will expect “{id\_type}:{value}”. Example: email:[test@test.com](mailto:test@test.com).
* custom\_auth\_proc is a procedure that must expect the parameter jwt\_payload. When validating the token, the system will pass to it the entire Payload of the token in JSON form.
* jwt\_throttle\_invalid\_count indicates how many failed attempts can the system take with the same Client ID in the token.
* jwt\_throttle\_time indicates how long the system will block requests from a Client ID after it exceeds the amount of allowed failed attempts.

How to Retrieve User Information

* If the token passed to the system is valid, we can access the user information in the token in the procedures used for the Endpoints.
* In the procedure used for the endpoint, call the procedure “ctrl::oauth::check\_auth\_header”
* If the token contained user information then the user\_id will be in $user\_info(user\_id) and the token used will be in $user\_info(token\_str)