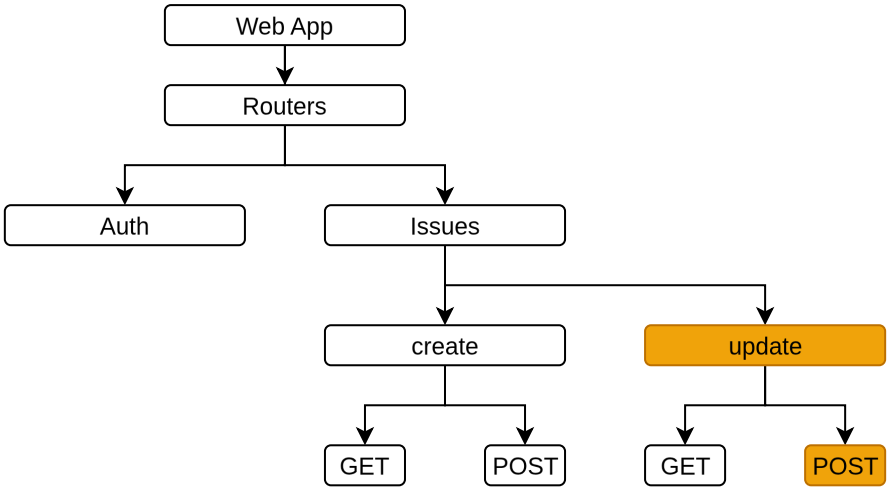
Contributors: Issues::Update::POST↗

YóUnăi ⊷ ames0k0
E-Mail↗ LinkedIn↗ Github↗

Abstract

Contributors is a web application designed to streamline open-source contribution by improving how developers discover and share coding tasks. Built to address the limitations of issue discovery on code hosting platforms with integrated issue trackers, it enables users to curate, tag, and submit contribution-friendly issues—helping developers quickly find clear, actionable opportunities to collaborate when they're ready to contribute.

Tech Stack: *Python*, *FastAPI*, *MongoDB*



Ticket

REQUEST :: EP=/issues/create/

Cookies		
NN	access_token	:: str

Params		
NN	issue_id	:: str

Form		
	url	:: str
NN	title	:: str
	description	:: str
NN	tags	:: str[,]
	labels	:: str[,]

CONTROLLER

Dependencies		
NN	c_users	:: DBCUsers
NN	c_issues	:: DBCIssues
NN	c_categories	:: DBCCategories
NN	current_user	:: UsersModel

Collection `issues`		
PK	_id	:: ObjectID
NN	url	:: str
NN	title	:: str
NN	description	:: str
NN	categories	:: json
NN	creation_dt	:: dt
NN	created_by	:: UserID

json		
NN	tags	:: Array[str]
NN	labels	:: Array[str]

Collection `categories`		
PK	_id	:: ObjectID
NN	identifier	:: str
NN	name	:: str
NN	issues_ids	:: Array[IssueId]

RESPONSE

InvalidTokenError		
EP	/issues/update/	
NN	401, detail	:: json

ValidationError		
EP	/issues/update/	
NN	422, detail	:: json

RedirectResponseError		
EP	/issues/update/	
NN	error_message	:: str

RedirectResponseSuccess		
EP	/issues/	
NN	error_message	:: str

Request

Params

```
if not P.issue_id then
    raise ValidationError
P.issue_id = strip_whitespaces(P.issue_id)
if len(P.issue_id) < 1 then
    raise ValidationError
```

IssueForm

```
if not {F.title, F.tags} then
    raise ValidationError
F.@ = strip_whitespaces(F.@)
if len(F.title) < 1 then
    raise ValidationError
if not len(comma_seperated_values(F.tags)) < 1 then
    raise ValidationError
```

Controller

Cookies

```
if not C.access_token then
    raise InvalidTokenError
```

Dependencies

```
D.c_* = db.get_collection()
D.current_user = Auth.get_current_user(C.access_token, D.c_users)
if not D.current_user then
    raise InvalidTokenError
```

BL

```
D.c_* = db.get_collection()
D.current_user = Auth.get_current_user(C.access_token, D.c_users)
if not D.current_user then
    raise InvalidTokenError
issue = D.c_issues.find_one(id=P.issue_id)
if not issue then
    redirect RedirectResponseError(error_message=NotExists)
if issue.created_by != D.current_user then
    redirect RedirectResponseError(error_message=PermissionsErrMsg)
if F.url then
    if not SupportedURLEnum(F.url) then
        redirect RedirectResponseError(error_message=NotSupportedUrl)
    issue = D.c_issues.find_one(id!=P.issue_id, url=F.url)
    if issue then
        redirect RedirectResponseError(error_message=Exists)
categories_to_delete = []
categories_to_update = []
created_categories = D.c_categories.find(issues_ids=P.issue_id)
for category in created_categories do
    if len(category.issues_ids) == 1 then
        categories_to_delete.append(category.id)
    else
        categories_to_update.append(category.id)
D.c_categories.delete_many({id: {$in: categories_to_delete}})
D.c_categories.update_many(filter=..., update={$pull: {issues_ids: P.issue_id}})
D.c_issues.update_one(filter=id=P.issue_id, update=F.@)
for tag_name in F.tags do
    tag = D.c_categories.find_one(identifier=tags, name=tag_name)
    if tag then
        D.categories.update_one(filter=..., update={$push: {issues_ids: P.issue_id}})
    else
        D.categories.insert_one(...)
for label_name in F.labels do
    tag = D.c_categories.find_one(identifier=labels, name=label_name)
    if tag then
        D.categories.update_one(filter=..., update={$push: {issues_ids: P.issue_id}})
    else
        D.categories.insert_one(...)
redirect RedirectResponseSuccess(error_message=Updated)
```