FREE eBook

LEARNING mongoose

Free unaffiliated eBook created from
Stack Overflow contributors.

#mongoose

Tabla de contenido

Sobre

Puede compartir este PDF con cualquier persona que crea que podría beneficiarse de él, descargue la última versión de: mongoose

Capítulo 1: Comenzando con la mangosta

Observaciones

> **Mongoose** es una herramienta de modelado de objetos de **MongoDB** diseñada para trabajar de forma asíncrona. ambiente.
>
> Todo en Mongoose comienza con un Esquema. Cada esquema se asigna a un MongoDB colección y define la forma de los documentos dentro de esa colección.

Mongoose hace que sea muy fácil trabajar con la base de datos MongoDB.

Podemos estructurar fácilmente nuestra base de datos usando esquemas y modelos, automatizar ciertas cosas cuando el registro se agrega o actualiza usando Middlewares/Hooks y obtener fácilmente los datos que necesitamos consultando nuestros modelos.

Links importantes

- Inicio rápido de mangosta
- Repositorio Mongoose GitHub
- Documentación de mangosta
- Complementos de mangosta

Versiones

Última versión: versión **4.6.0** lanzada el **2 de septiembre de 2016**

Todas las versiones se pueden encontrar en https://github.com/Automattic/mongoose/blob/master/History.md

Versión	Fecha de lanzamiento
1.0.1	2011-02-02
1.1.6	2011-03-22
1.3.0	2011-04-19
1.3.1	2011-04-27
1.3.4	2011-05-17
1.4.0	2011-06-10
1.5.0	2011-06-27
1.6.0	2011-07-07
2.0.0	2011-08-24

Versión	Fecha de lanzamiento
2.3.4	2011-10-18
2.5.0	2012-01-26
3.0.0	2012-08-07
3.1.2	2012-09-10
3.2.0	2012-09-27
3.5.0	2012-12-10
3.5.6	2013-02-14
3.6.0	2013-03-18
3.6.5	2013-04-15
3.8.0	2013-10-31
3.8.10	2014-05-20
3.8.15	2014-08-17
4.0.0	2015-03-25
4.0.6	2015-06-21
4.1.0	2015-07-24
4.2.0	2015-10-22
4.2.10	2015-12-08
4.3.5	2016-01-09
4.4.0	2016-02-02
4.4.4	2016-02-17
4.4.8	2016-03-18
4.4.13	2016-04-21
4.4.18	2016-05-21
4.5.0	2016-06-13
4.5.5	2016-07-18

Versión	Fecha de lanzamiento
4.5.8	2016-08-01
4.5.9	2016-08-14
4.5.10	2016-08-23
4.6.0	2016-09-02

Ejemplos

Instalación

Instalar **mongoose** es tan fácil como ejecutar el comando npm

```
npm instalar mongoose --save
```

Pero asegúrese de haber instalado MongoDB para su sistema operativo o tener acceso a una base de datos MongoDB.

Conexión a la base de datos MongoDB:

1. Importar mangosta a la aplicación:

```
importar mangosta de 'mangoose';
```

2. Especifique una biblioteca Promise:

```
mangosta.Promesa = global.Promesa;
```

3. Conéctese a MongoDB:

```
mongoose.connect('mongodb://127.0.0.1:27017/base de datos');

/* El formato de conexión de Mongoose se parece a esto */ mongoose.connect('mongodb://NOMBRE
DE USUARIO:CONTRASEÑA@HOST::PUERTO/NOMBRE_BASE DE DATOS');
```

Nota:

- De forma predeterminada, mongoose se conecta a MongoDB en el puerto 27017, que es el puerto predeterminado utilizado por MongoDB.
- Para conectarse a MongoDB alojado en otro lugar, use la segunda sintaxis. Ingrese el nombre de usuario, la contraseña, el host, el puerto y el nombre de la base de datos de MongoDB.

El puerto MongoDB es 27017 por defecto; use el nombre de su aplicación como el nombre de la base de datos.

Conexión con opciones y callback

Mongoose connect tiene 3 parámetros, uri, opciones y la función de devolución de llamada. Para usarlos, vea el ejemplo a continuación.

```
var mangosta = require('mangosta');

var uri = 'mongodb: // localhost: 27017 / DBNAME';

var opciones = { usuario:
        'usuario1', paso: 'paso'

 }

mongoose.connect(uri, options, function(err){ if (err) throw err; // si no hay
        error == conectado

 });
```

Lea Empezando con la mangosta en línea : https://riptutorial.com/es/mangoose/topic/1168/empezando-con-la-mangosta

Capítulo 2: Middleware de Mangosta

Observaciones

En mongoose, los **Middlewares** también se denominan ganchos previos y posteriores .

Hay dos tipos de middleware

Ambos middleware admiten enlaces **previos** y **posteriores .**

1. **Middleware de documento**

   Es compatible con las funciones de documento inicializar, validar, guardar y retirar

2. **Consulta de software intermedio**

   Es compatible con las funciones de consulta count, find, findOne, findOneAndRemove, findOneAndUpdate, insertMany y actualice.

Ganchos antes y después

Hay dos tipos de ganchos **Pre**

1. **serie**

   Como sugiere el nombre, se ejecuta en orden de serie, es decir, uno tras otro.

2. **paralelo**

   El middleware paralelo ofrece un control de flujo más detallado y el método enganchado no es ejecutado hasta que termine es llamado por todo el middleware paralelo.

El Middleware **posterior** se ejecuta después del método enganchado y todos sus el middleware ha sido precompletados .

Los **métodos enganchados** son las funciones admitidas por el middleware de documentos. init, validar, guardar, retirar

Ejemplos

Middleware para cifrar la contraseña del usuario antes de guardarla

Este es un ejemplo de **Middleware de documentos en serie**

En este ejemplo, escribiremos un middleware que convertirá la contraseña de texto sin formato en un hash

contraseña antes de guardarla en la base de datos.

Este middleware se activará automáticamente al crear un nuevo usuario o actualizar los detalles del usuario existente.

NOMBRE DE ARCHIVO : Usuario.js

```
// vamos a importar mangosta primero importar
mangosta desde 'mangoose'

// vamos a crear un esquema para el modelo de usuario
const UserSchema = mangosta.Schema(
    {
        nombre: cadena,
        correo electrónico: {tipo: cadena, minúsculas: verdadero, requisito: verdadero}, contraseña:
        cadena,},

);


/**
  * Este es el middleware, se llamará antes de guardar cualquier registro */

UserSchema.pre('guardar', función(siguiente) {

    // comprobar si la contraseña está presente y se modifica. if ( esta.contraseña
    && esta.esModificada('contraseña') )                                    {

        // llame a su método hashPassword aquí, que devolverá la contraseña cifrada. esta.contraseña = hashPassword(esta.contraseña);


    }

    // todo está hecho, así que llamemos a la próxima devolución de llamada. Siguiente();


});


// vamos a exportarlo, para que podamos importarlo en otros archivos. exportar
mongoose.model predeterminado ('Usuario', UserSchema);
```

Ahora, cada vez que guardamos un usuario, este middleware verificará si la contraseña está **configurada** y **modificada** (esto es para que no hagamos hash de la contraseña de los usuarios si no se modifica).

NOMBRE DE ARCHIVO : App.js

```
importar expreso de 'expreso'; importar mangosta
de 'mangoose';

// vamos a importar nuestro modelo de usuario
import User from './User';

// conectarse a MongoDB
mongoose.Promise = global.Promise;
mongoose.connect('mongodb://127.0.0.1:27017/base de datos');
```

```
dejar aplicación = express(); /* ...
expresar middlewares aquí .... */


aplicación.post( '/', (requerido, res) => {

    /*
        req.cuerpo = {
            nombre: 'John Doe',
            correo electrónico: 'john.doe@gmail.com',
            contraseña: '!trump'
        }
      */

    // validar los datos POST

    let nuevoUsuario = nuevo Usuario({
        nombre: req.body.name,
        correo electrónico: req.body.email,
        contraseña: req.body.password,
    });

    nuevoUsuario.guardar( ( error, registro ) => {
        if (error)

            { res.json({ mensaje: 'error',
                descripción: 'ocurrió algún error al guardar al usuario en la base de datos'.
            });
        } más {
            res.json({mensaje:
                'éxito', descripción: 'los detalles
                del usuario se guardaron correctamente',
                usuario: registro
            });
        }
    });

});



let server = app.listen( 3000, () => {
        console.log(`Servidor ejecutándose en http://localhost:3000` );
    }
);
```

Lea Middleware de mangosta en línea: https://riptutorial.com/es/mongoose/topic/6646/middleware de mangosta

Capítulo 3: Población de mangostas

Sintaxis

1. Model.Query.populate(ruta, [seleccionar], [modelo], [coincidencia], [opciones]);

Parámetros

Detalles de parámetro	
sendero	**Cadena :** la clave de campo que se completará
select **Object, String** - Selección de campo para la consulta de población.	
modelo **Modelo** - Instancia del modelo referenciado	
Coincidencia de **objetos :** condiciones de relleno	
Opciones **Objeto** - Opciones de consulta	

Ejemplos

Poblar simple

Mongoose poblar se utiliza para mostrar datos de documentos de referencia de otras colecciones.

Digamos que tenemos un modelo de Persona que tiene documentos de referencia llamados Dirección.

modelo de persona

```
var Persona = mangosta.model('Persona', {
        fname: Cadena,
        mname: Cadena,
        lname: Cadena,
        dirección: {tipo: Schema.Types.ObjectId, ref: 'Address'}
});
```

Modelo de dirección

```
var Dirección = mongoose.model('Dirección', { houseNum: Cadena,
        calle: Cadena, ciudad: Cadena, estado: Cadena, país: Cadena




});
```

Para completar la Dirección dentro de la Persona usando su ObjectId, usando digamos findOne(), use la función populate() y agregue la dirección de la clave de campo como el primer parámetro.

```
Person.findOne({_id: req.params.id}) .populate('address') //
        <- usa la función populate() .exec(function(err, person) { // hacer algo. // variable ` person`
        contiene los datos completos finales


        });
```

O

```
Person.findOne({_id: req.params.id}, function(err, person) { // hacer algo // variable `person` contiene
        los datos completos finales

})
 .poblar('dirección');
```

La consulta anterior debe producir el documento a continuación.

Persona Doc

```
{
        "_id":"123abc",
        "fname":"John",
        "mname":"Kennedy",
        "lname":"Doe",
        "address":"456def" // <- ID de dirección
}
```

Documento de dirección

```
{
        "_id":"456def",
        "houseNum":"2",
        "street":"Street 2", "city":"Ciudad
        de los muertos", "state":"AB", "country:"PH"


}
```

Documento poblado

```
 {
        "_id":"123abc",
        "fname":"John",
        "mname":"Kennedy",
        "lname":"Doe", "address":
        { "_id":"456def", "houseNum":"
                2", "calle":"Calle 2",
                "ciudad":"Ciudad de los
                muertos", "estado":"AB", "país:"PH"
```

```
        }
}
```

Descuide algunos campos

Supongamos que **no desea que** los campos houseNum y street estén en el campo de dirección del documento completo final, use populate() de la siguiente manera:

```
Persona.findOne({_id: req.params.id})
        .populate('address', '-houseNum -street') // observe el `-` .exec(function(err, person) { // hacer símbolo
        algo. // la variable `person` contiene los datos completos finales


        });
```

O

```
Person.findOne({_id: req.params.id}, function(err, person) { // hacer algo // variable `person` contiene
        los datos completos finales


}) .populate('dirección', '-houseNum -street'); // nota el `-` símbolo
```

Esto producirá el siguiente documento completo final,

Documento poblado

```
  {
        "_id":"123abc",
        "fname":"Juan",
        "mname":"Kennedy",
        "lname":"Doe", "dirección":
        { "_id":"456def", "ciudad":"
                Ciudad de los muertos",
                "estado":"AB", "país:"PH"


        }
}
```

Rellene solo unos pocos campos

Si **solo desea** los campos houseNum y street en el campo de dirección en el documento completo final, use la función populate() de la siguiente manera en los dos métodos anteriores,

```
Persona.findOne({_id: req.params.id})
        .populate('address', 'houseNum street') .exec(function(err,
        person) { // hacer algo. // variable `person` contiene los datos
                completos finales

        });
```

O

```
Person.findOne({_id: req.params.id}, function(err, person) { // hacer algo // variable `person` contiene
        los datos completos finales

})
 .populate('dirección', 'núm. de casa calle');
```

Esto producirá el siguiente documento completo final,

Documento poblado

```
  {
        "_id":"123abc",
        "fname":"Juan",
        "mname":"Kennedy",
        "nombre": "Hacer",
        "Dirección":{
                "_id": "456def", "Número
                de casa": "2",
                "calle":"Calle 2"
        }
}
```

Población anidada

Digamos que tiene un esquema de usuario , que contiene `nombre` , número de contacto, dirección y amigos.

```
var UserSchema = nueva mangosta.Schema({
        nombre: cadena,
        número de contacto: número,
        dirección: cadena, amigos: [{

                tipo: mangosta.Schema.Types.ObjectId,
                ref : Usuario
        }]
});
```

Si desea encontrar un usuario, sus **amigos** y los **amigos de sus amigos,** debe realizar la población en 2 niveles, es decir, **Población anidada.**

Para buscar amigos y amigos de amigos:

```
User.find({_id : userID}) .populate({ ruta :
        'amigos', poblar :

                                { ruta : 'amigos'}//para encontrar amigos de amigos
        });
```

Todos los parámetros y opciones de poblar también se pueden usar dentro de poblar anidados para obtener el resultado deseado.

Del mismo modo, puede completar más niveles según sus requisitos.

No se recomienda hacer población anidada por más de 3 niveles. En caso de que necesites hacer

relleno anidado para más de 3 niveles, es posible que deba reestructurar su esquema.

Lea Población de mangostas en línea: https://riptutorial.com/es/mangoose/topic/2616/poblacion de mangostas

Capítulo 4: Población de mangostas

Sintaxis

- Query.populate(ruta, [seleccionar], [modelo], [coincidencia], [opciones])

Parámetros

Parámetro	Explicación
sendero	**<Objeto, Cadena>** ya sea la ruta para completar o un objeto que especifica todos los parámetros
[Seleccione]	**<Objeto, Cadena>** Selección de campo para la consulta de población (puede usar '-id' para incluir todo menos el campo id)
[modelo]	**<Modelo>** El modelo que desea usar para la población. Si no se especifica, poblar buscará el modelo por el nombre en el campo de referencia del esquema.
[juego]	**<Objeto>** Condiciones para la población
[opciones]	**<Objeto>** Opciones para la consulta de población (ordenar, etc.)

Ejemplos

Un ejemplo simple de población de mangostas

.populate() en Mongoose le permite completar una referencia que tiene en su colección o documento actual con la información de esa colección. Lo anterior puede sonar confuso, pero creo que un ejemplo ayudará a aclarar cualquier confusión.

El siguiente código crea dos colecciones, User y Post:

```
var mangosta = require('mangosta'), Esquema =
    mangosta.Esquema

var esquema de usuario = esquema ({
    nombre: cadena,
    edad: número,
    publicaciones: [{ tipo: Schema.Types.ObjectId, ref: 'Post' }] });


var PostSchema = Esquema({
    usuario: { tipo: Schema.Types.ObjectId, ref: 'User' }, title: String, content: String });
```

```
var Usuario = mongoose.model('Usuario', userSchema); var Post =
mongoose.model('Post', postSchema);
```

Si quisiéramos completar todas las publicaciones de cada usuario cuando .encontremos ({}) a todos los usuarios, podríamos hacer lo siguiente:

```
Usuario
    .encontrar({})
    .populate('posts') .exec(function(err,
    users) { if(err) console.log(err); //esto registrará
        a todos los usuarios con cada una de sus
        publicaciones else console.log(users);

    })
```

Lea Población de mangostas en línea: https://riptutorial.com/es/mangoose/topic/6578/poblacion de mangostas

Capítulo 5: mongoose pre y post middleware (ganchos)

Ejemplos

software intermedio

El middleware (también llamado pre y post hooks) son funciones a las que se les pasa el control durante la ejecución de funciones asincrónicas. El middleware se especifica en el nivel de esquema y es útil para escribir complementos. Mongoose 4.0 tiene 2 tipos de middleware: middleware de documentos y middleware de consultas. El middleware de documentos es compatible con las siguientes funciones de documentos.

- iniciar
- validar
- ahorrar
- retirar

El middleware de consulta es compatible con las siguientes funciones de modelo y consulta.

- contar
- buscar
- buscar uno
- buscar uno y quitar • buscar uno y actualizar • actualizar

Tanto el middleware de documentos como el middleware de consultas admiten ganchos previos y posteriores.

Pre

Hay dos tipos de ganchos previos, en serie y en paralelo.

De serie

El middleware serial se ejecuta uno tras otro, cuando cada middleware llama a continuación.

```
var esquema = nuevo esquema (..);
esquema.pre('guardar', función(siguiente) {
    // hacer cosas
    Siguiente();
});
```

Paralela

El middleware paralelo ofrece un control de flujo más detallado.

var esquema = nuevo esquema (..);

```
// `true` significa que se trata de un middleware paralelo. **Debe** especificar `true` // como segundo parámetro si desea
utilizar middleware paralelo. esquema.pre('guardar', verdadero, función(siguiente, hecho) {

    // llamar a next inicia el siguiente middleware en paralelo
    Siguiente();
    setTimeout(hecho, 100);
});
```

El método enganchado, en este caso save, no se ejecutará hasta que cada middleware llame a done.

Publicar middleware

el middleware posterior se ejecuta después de que el método enganchado y todo su middleware previo se hayan completado. el middleware posterior no recibe directamente el control de flujo, por ejemplo, no se le pasan devoluciones de llamada next o done. los ganchos posteriores son una forma de registrar detectores de eventos tradicionales para estos métodos.

```
schema.post('init', function(doc) { console.log('%s ha sido
    inicializado desde la base de datos', doc._id); }); esquema.post('validar', función(doc) {


    console.log('%s ha sido validado (pero aún no guardado)', doc._id); }); esquema.post('guardar', función(doc) {


    console.log('%s ha sido guardado', doc._id); }); esquema.post('eliminar',
función(doc) {

    console.log('%s ha sido eliminado', doc._id); });
```

Lea mongoose pre y post middleware (ganchos) en línea: https://riptutorial.com/es/mongoose/topic/4131/mongoose-pre-and-post-middleware--hooks

Capítulo 6: Consultas de mangosta

Introducción

Mongoose es un controlador de Node.JS para MongoDB. Proporciona ciertos beneficios sobre el controlador MongoDB predeterminado, como agregar tipos a esquemas. Una diferencia es que algunas consultas Mongoose pueden diferir de sus equivalentes MongoDB.

Ejemplos

Buscar una consulta

Importe un modelo Mongoose (consulte el tema "Esquemas Mongoose")

```
var Usuario = require("../models/user-schema.js")
```

El método findOne devolverá la primera entrada en la base de datos que coincida con el primer parámetro. El parámetro debe ser un objeto donde la clave es el campo a buscar y el valor es el valor que debe coincidir. Esto puede usar la sintaxis de consulta de MongoDB, como el operador de punto (.) para buscar subcampos.

```
Usuario.findOne({"nombre": "Fernando"}, función(err, resultado){
        si (err) tirar err; //Hubo un error con la base de datos. if(!result) console.log("Nadie se llama
        Fernando."); //La consulta no encontró resultados. else { consola.log(resultado.nombre); //Imprime "Fernando"


        }
}
```

Lea Consultas de mangosta en línea: https://riptutorial.com/es/mongoose/topic/9349/consultas-de-mangoose

Capítulo 7: Esquemas de mangosta

Ejemplos

Esquema básico

Un esquema de usuario básico:

```
var mangosta = require('mangosta');

var userSchema = nueva mangosta.Schema({
        nombre: Cadena,
        contraseña: Cadena, edad:
        Número, creado: {tipo:
        Fecha, predeterminado: Fecha.ahora}
});

var Usuario = mongoose.model('Usuario', userSchema);
```

Tipos de esquema.

Métodos de esquema

Los métodos se pueden establecer en esquemas para ayudar a hacer cosas relacionadas con ese(s) esquema(s) y mantenerlos bien organizados.

```
userSchema.methods.normalize = function() { this.name =
        this.name.toLowerCase();
};
```

Ejemplo de uso:

```
erik = nuevo usuario ({
        'nombre': 'Erik',
        'contraseña': 'contraseña'
});
ciruela.normalizar();
plum.save();
```

Estática de esquema

Los esquemas estáticos son métodos que un modelo puede invocar directamente (a diferencia de los métodos de esquema, que deben ser invocados por una instancia de un documento Mongoose). Usted asigna un Estático a un esquema agregando la función al objeto estático del esquema .

Un caso de uso de ejemplo es para construir consultas personalizadas:

```
userSchema.statics.findByName = función (nombre, devolución de llamada) {
```

```
        devuelve this.model.find ({ nombre: nombre }, devolución de llamada);
}

var Usuario = mongoose.model('Usuario', userSchema)

User.findByName('Kobe', function(err, documentos) {
        consola.log(documentos)
})
```

Esquema de GeoObjetos

Un esquema genérico útil para trabajar con objetos geográficos como puntos, cadenas lineales y polígonos. Tanto **Mongoose** como **MongoDB** son compatibles **con Geojson.**

Ejemplo de uso en **Node/Express:**

```
var mangosta = require ('mangosta'); var Schema =
mangosta.Schema;

// Crea un esquema GeoObject. var myGeo= nuevo
esquema({
                                        nombre: {tipo: Cadena},
                                        geo:        {
                                                escribe :   {
                                                        tipo: Cadena,
                                                        enumeración: ['Punto', 'LineString', 'Polígono']
                                                },
                                                coordenadas: Matriz
                                          }
});

//Índice de 2dsphere en el campo geográfico para trabajar con consultas geoespaciales myGeo.index({geo :
'2dsphere'}); módulo.exportaciones = mongoose.model('myGeo', myGeo);
```

Guardar la hora actual y la hora de actualización

Este tipo de esquema será útil si desea realizar un seguimiento de sus elementos por tiempo de inserción o tiempo de actualización.

```
var mangosta = require ('mangosta'); var Schema =
mangosta.Schema;

// Crea un esquema de usuario.
var usuario = nuevo esquema ({ nombre:
                                        { tipo: Cadena }, { tipo: Entero}, { tipo:
                                        años :      Cadena }, created_at: {tipo:
                                        sexo:       Fecha, predeterminado:
                                        Date.now}, updated_at: {tipo: Fecha, predeterminado: Fecha.ahora}

});

// Establece el parámetro created_at igual a la hora actual user.pre('save', function(next){ now = new
Date(); this.updated_at = now;
```

```
        if(!this.created_at) {
            this.created_at = ahora
        }
        Siguiente();
});

módulo.exportaciones = mangosta.modelo('usuario', usuario);
```

Lea Esquemas de mangosta en línea: https://riptutorial.com/es/mongoose/topic/2592/esquemas de mangosta

Capítulo 8: Esquemas de mangosta

Ejemplos

Creación de un esquema

```
var mangosta = require('mangosta');

//asumir que los esquemas del jugador y del tablero ya están hechos var Player =
mongoose.model('Player'); var Tablero = mongoose.model('Tablero');


//Cada clave en el esquema está asociada con el tipo de esquema (es decir, cadena, número, fecha, etc.) var gameSchema = new
mongoose.Schema({
    nombre: Cadena,
    jugadores: [{
                tipo: mangosta.Schema.Types.ObjectId, ref: 'Jugador'

    }],
    anfitrión: {
        tipo: mangosta.Schema.Types.ObjectId, ref: 'Jugador'

    },
    junta: {
        tipo: mangosta.Schema.Types.ObjectId,
        ref: 'Tablero'
    },
    activo: {
        tipo: booleano,
        predeterminado: verdadero
    },
    estado: {
        tipo: cadena,
        enumeración: ['decisión', 'ejecutar', 'esperando'], predeterminado:
        'esperando'
    },
    numFlags: { tipo:
        Número, enumeración:
        [1,2,3,4]
    },
    esGanado: {
        tipo: booleano,
        predeterminado: falso
    }
});

mongoose.model('Juego', gameSchema);
```

Lea Esquemas de mangosta en línea: https://riptutorial.com/es/mongoose/topic/6622/esquemas de mangosta

Créditos

S. No	Capítulos	Colaboradores
1	Empezar con mangosta	4444, CENT1PEDE, Comunidad, Delapouite, duplicador, jisoo, Usuario aleatorio, zurfyx
2	Mangosta software intermedio	Delapouite, Usuario aleatorio
3	Mangosta Población	CENT1PEDE, Porcelana, Medet Tleukabiluly, Ravi Shankar
4	mangosta pre y post middleware (manos)	Naeem Shaikh
5	consultas de mangosta	Gibryon Bhojraj
6	esquemas de mangosta	AndreaM16, ian, zurfyx