***Solo es obligatoria la entrega de la parte del Student pero pidieron que el resto esté listo como para poder agregarlo si fuese necesario.***

El usuario se logea:

1. Es Student
   1. Campus
      1. Puede ver sus materias y los libros de cada una, se deberá indicar que libros tiene.
      2. Inscribirse a materias
   2. Ecommerce (Debe tener cuenta bancaria)
      1. Puede agregar y sacar libros del carrito
      2. Checkout -> Factura
      3. Historial de compras
      4. Consulta soporte
   3. Banco
      1. Crear cuenta
      2. Deposito -> Comprobante
      3. Retiro -> Comprobante
      4. Historial de transacciones
   4. Editar información
      1. Cambiar contraseña
      2. Cambiar nombre (no username)
      3. Cambiar número de teléfono
2. Es Admin
   1. Crear cuenta de support
   2. Crear cuenta de proveedor
   3. Dar de baja support
   4. Dar de baja proveedor
   5. Dar de baja student
   6. Agregar materias y sus BookInformation (Puede incluir BookInformation ya existentes)
   7. Agregar Carrera y sus materias (Puede incluir Materias ya existentes)
   8. Ver cuenta de universidad e historial
3. Es Provider
   1. Agregar ProductBook de un BookInformation existente
   2. Dar de baja ProductBook
4. Es SupportStaff
   1. Contestar consultas

Notas:

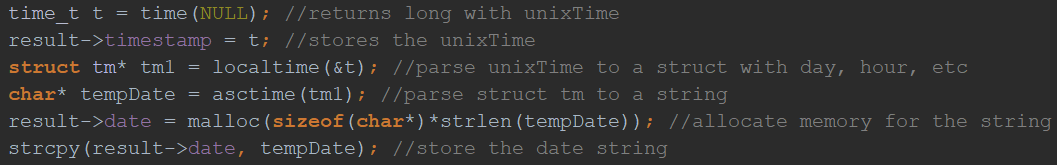
Cuando un usuario va a ingresar, pone su usuario y contraseña, con esto se busca en la base de datos en la lista de UserLog para ver que userType y userID está relacionado con ese userName, luego al saber el tipo se utilizan los respectivos gets para encontrar el usuario y comparar su contraseña. ***Este proceso se realiza solo al loggearse, las demás estructuras saben el id del usuario.***

Todos los que sean usuarios van a tener los atributos que tiene usuario. Por ejemplo, el Student, además de tener carrera, books, etc tiene todos los atributos de User. **No existe un struct user por sí solo.**

Todo usuario se crea sin el Id seteado, este se setea cuando se lo agrega a la UserDatabase.

Los usuarios van a estar almacenados únicamente en la UserDatabase para evitar duplicación. Si algún modulo necesita acceder a algún usuario será a través de UserDatabase.

“Timestamp” es un long conteniendo el unixTime del momento en que algo fue creado, con esto se puede autogenerar un string conteniendo la fecha y se guardan de la siguiente manera:



Payment va a ser un struct conteniendo el Invoice y el PaymentMethod, el PaymentMethod va a tener un int señalando si es transferencia, depósito o tarjeta, el int de paymentData por lo tanto significara una cosa distinta para cada tipo.

Una vez que se genera el payment, se pasa a realizar una transacción en el banco, se genera un retiro de la cuenta del Student y un ingreso a la cuenta de la universidad, conectada al Admin.

Cosas que a las que no encuentro utilidad:

* Contract
* ProviderLog
* Receipt
* ReceiptProductContent
* Selecciona
* Shipping
* Address
* RegisteredUserInvoice