## Laboration Runt Optimering Av Konto skapande

### Gammal Strategi | “OOP” implementation

Kalla på Save() metoden för typ UserServer.

Save metoden öppnar accounts.yaml filen som innehåller alla konton, använder go-builltin append för att lägga till nya kontot, sparar om filen på nytt.

Positivt

* En fil som innehåller alla konton
* Enkelt att kolla om en användare existerar

Negativt

* Kräver att öppna hela accounts.yaml filen bara för att lägga till en användare
* Kräver att öppna hela accounts.yaml filen bara för att läsa eller hämta en användare
* Processen blir söligare desto mer konton som finns eftersom mer måste laddas

### Ny Strategi | “Account Writer” implementation

Alla konton har sin egna {username}.yaml fil.

Functionen för att skapa användare kallas bara en gång vid init (skapar separat go routine), och körs sedan i bakgrunden under hela programmets längd. Den väntar på SigReg channel som innehåller UserServer data.

Kallande funktion väntar på att få OK signal tillbaka via SigOk från Writer

{username}.yaml fil skapas från UserServer.Username och UserServer kodas in till den.

Positivt

* Renare Kod
* Behöver inte röra övriga existerande konton för att hantera eller lägga till ett specifikt konto
* Enklare att ta bort konton
* Högre prestanda

Negativt

* Pointers till SigReg och SigOk måste importeras till alla metoder
* Svårare fel hantering

### Resultat

|  |  |  |
| --- | --- | --- |
| **Test Beskrivning** | **OOP Implementering** | **Writer Implementering** |
| Tid för 1000 konton att skapas | 10s | 2.7% CPU | 0.06s | 12% CPU |
| Tid för 10 konton att skapas | 0.01s | 2.9% CPU | 0.003s | 11% CPU |

i7 6600U 4x @3.1GHz, Samsung EVO SSD

Disken är vad som begränsar hastigheten, att inte behöva läsa in kontoinformation för att skapa nya leder till gigantiska prestandaskillnader. CPU användning är högre eftersom CPU kan göra mer när Disken inte begränsar hastighet lika mycket.