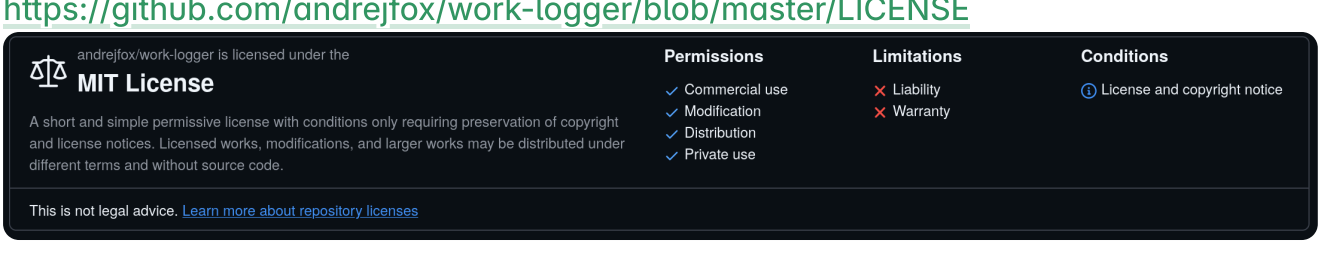
Osnutek

V repo-tu je vsa koda, kratek opis in navodila za pogon aplikacije v angleščini:
https://github.com/andrejfox/work-logger

Projekt je pod MIT licenco:
https://github.com/andrejfox/work-logger/blob/master/LICENSE



Prvotna motivacija za projekt je bila lastna lenoba za zapisovanje ur dela. Po parih incidentih, ko sem pozabil zapisati ure, sem se odločil, da moram nekaj spremeniti. Tako sem dobil idejo za Work Logger-ja, aplikacija za hitro in učinkovito beleženje ur na osnovi jave.
Ravno, ko sem aplikacijo končal in jo začel uporabljati pa se je moj delodajalec odločil, da moramo vse pisati v excel file ;-;

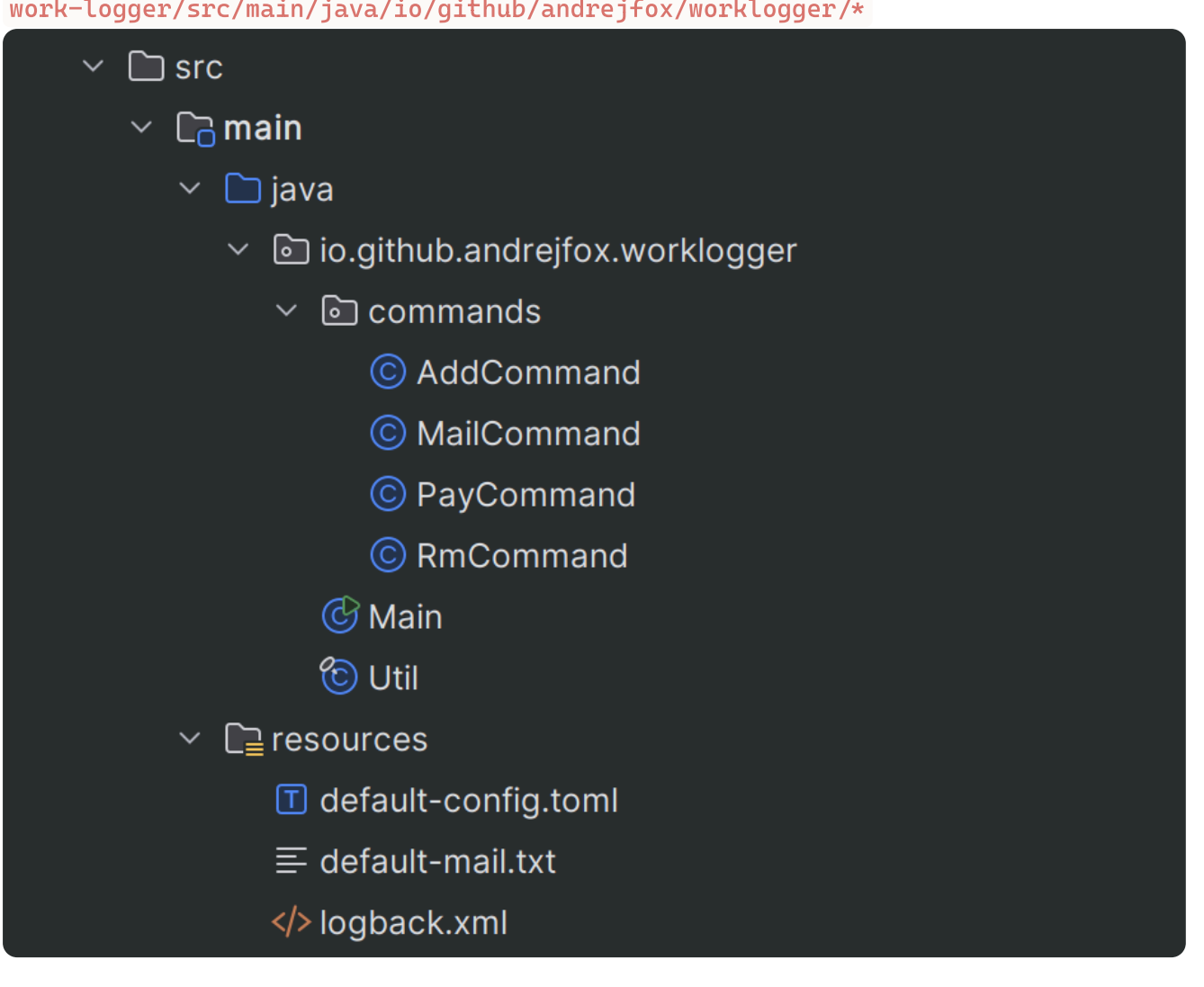
Cilji:

1. Delovati moda dovolj dobro, da jo bom sam uporabljal
2. Preprost način zaganjanja, da si lahko kdorkoli self host-a aplikacijo
3. Funkcionalnosti:
   - Beleženje ur
   - Brisanje ur
   - Formatiranje podatkov v e-mail za delodajalca
   - Beleženje ne/plačanih mescev
   - (Statistika čez določen čas)
4. Dostopnost na vseh platformah (pc, telefon,...)
5. 24/7 dostopnost

Izbira grafike
Prva stvar s katero sem se imel prave težave je izbira grafičnega vmesnika, ki bi mi omogočal delovanje na večih platformah. Prva izbira je bil swift, kak dan po uporabi swift-a pa sem se spomnil, da pogosto uporabljam nekaj kar bi lahko uporabil kot grafični vmesnik, Discord! Ne samo, da poenostavi grafiko (katere nisem največji oboževalec), omogoča mi, da se poigram z Api-ji in se naučim še malo več o njih.

Shranjevanje podatkov
Razmišljal sem ali naj podatke shranjujem v PB ali samo v navadnih datotekah (npr.: txt, json,...). Na koncu sem se odločil za shranjevanje v .json saj je precej preprosto in mi bo omogočilo, da se osredotočim na komunikacijo z Discord api-jem in logiko aplikacije. (potencialna izboljšava, da se podatke shranjuje v PB)

Struktura projekta
IDE, ki sem ga uporabil je Intellij community edition by JetBrains.
Poskušal sem se držati standardni obliki snovanja javanskih aplikacij, glavni deli kode se nahajajo pod:
`work-logger/src/main/java/io/github/andrejfox/worklogger/*`



Knjižnice
Najpomembnejše so:
JDA (Java Discord Api),
gson (formatiranje .json),
toml4j (formatiranje .toml (config files)),
(java standard library)

```
dependencies {
    testImplementation(platform("org.junit:junit-bom:5.9.1"))
    testImplementation("org.junit.jupiter:junit-jupiter")
    implementation("net.dv8tion:JDA:5.0.0-beta.24")
    implementation("ch.qos.logback:logback-classic:1.4.14")
    implementation("com.google.code.gson:gson:2.10.1")
    implementation("com.moandjiezana.toml:toml4j:0.7.2")
}
```

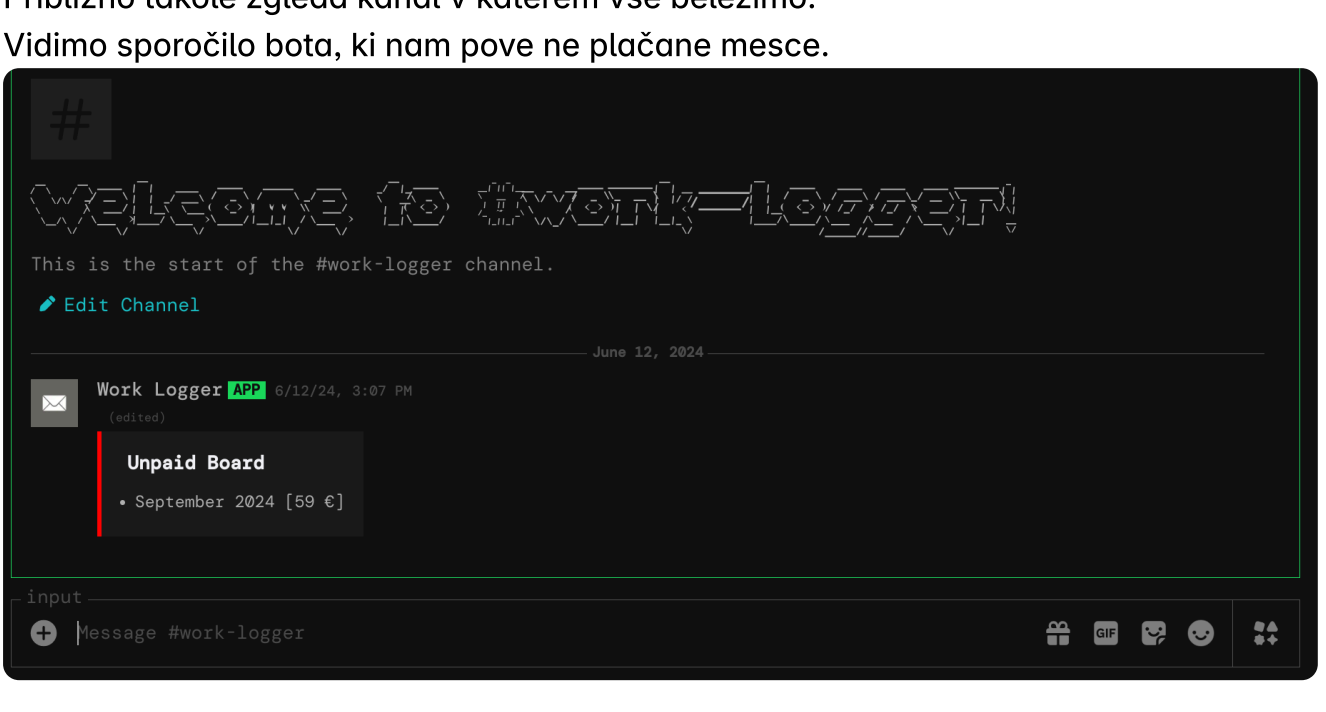
Privzete nastavitvene datoteke
Privzete nastavitvene datoteke se same ustvarijo ob prvem zagonu.
Za nastavljanje splošnih nastavitev se vse nahaja v config.toml
`botToken` ← Nastavljanje token-a za api
`channel` ← ID kanala v Discordu kamor ženimo, da bot piše
`messageID` ← ID sporočila za poplačila (to se bo smo izpolnilo ob prvem zagonu)
`languageTag` ← Nastavitev jezika za pravilno formatiranje datumov
`currency` ← Znak, ki bo uporabljen za valuto
`paymentTypes` ← array struct-ov, ki definirajo postavke { <ime>, <multiplier> }

```
botToken = "" #Bot token goes here
channel = 0 #Channel id goes here
messageID = 0 #Message will be here
languageTag = "en"
currency = "€"
paymentTypes = [
    { tag = "<name>", type = 10 }
]
```

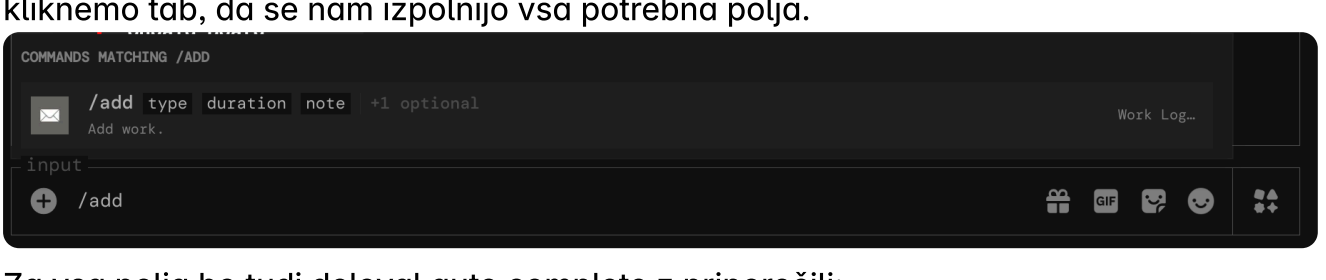
Za formatiranje e-maila pa se nastavitev nahaja v default-mail.txt, kar je v {} bo program dopolnil z podatki poljubnega mesca ob izvršitvi /mail komande.

```
Dear Sir/Madam,
Sending hours for {MONTH}.

{WORK_DATA}

In total:
{DATA_SUM}

Regards,
Name Surname
```

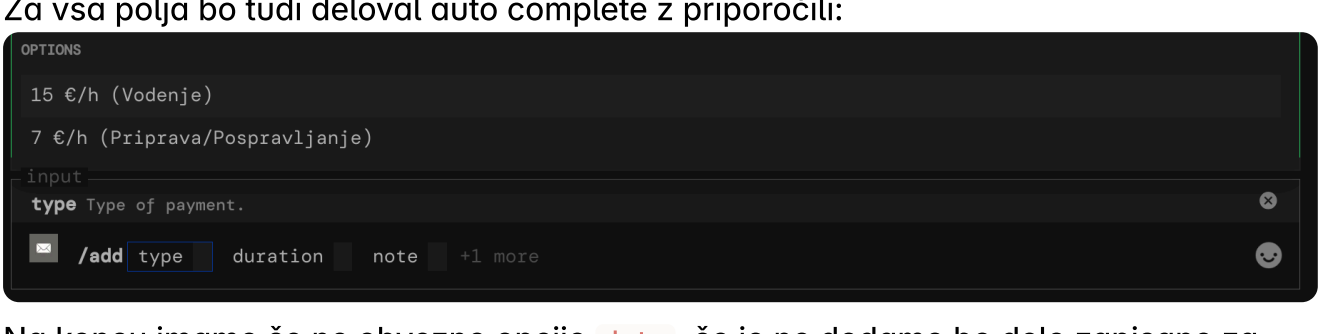
Uporaba
Približno takole zgleda kanal v katerem vse beležimo.
Vidimo sporočilo bota, ki nam pove ne plačane mesce.



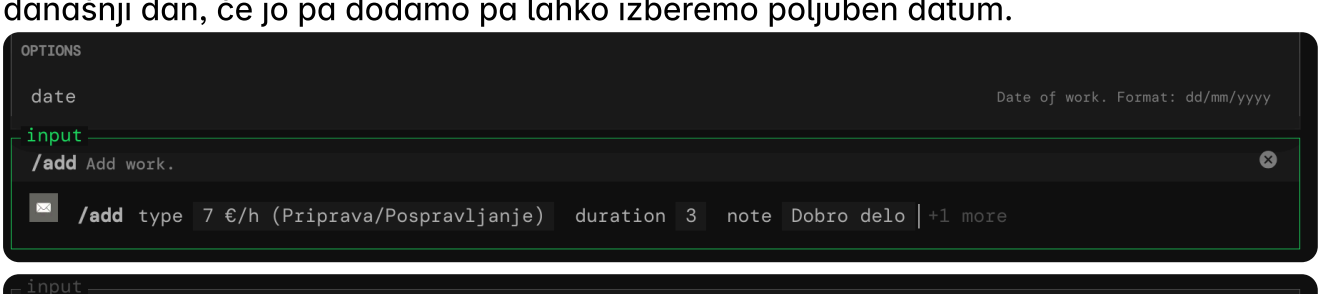
/add
Komandam sem implementiral auto complete tako da samo napišemo /add in kliknemo tab, da se nam izpolnijo vsa potrebna polja.

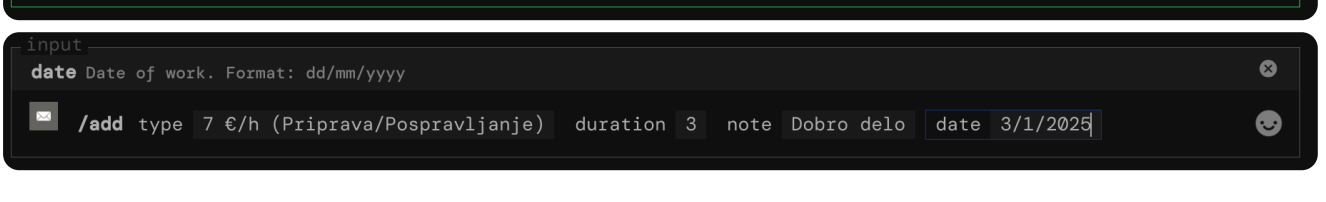


Za vsa polja bo tudi deloval auto complete z priporočili:

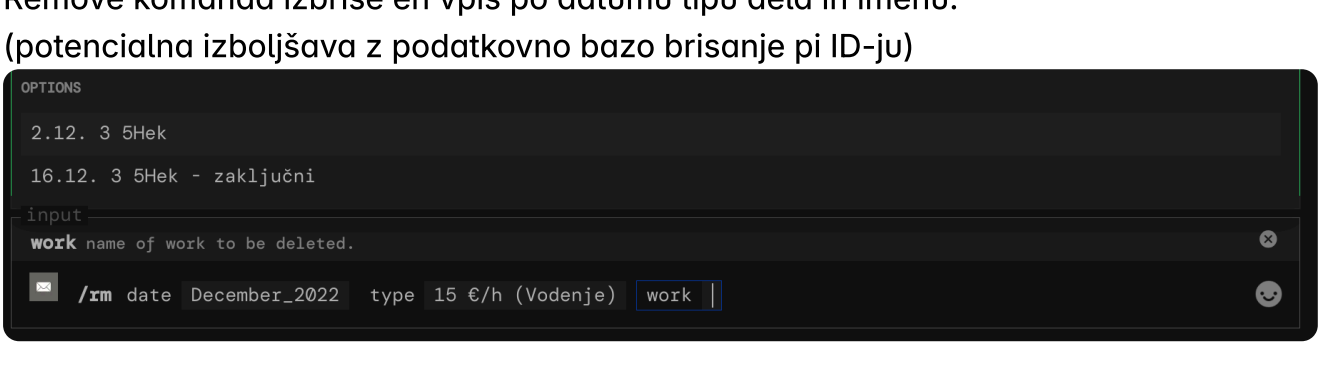


Na koncu imamo še ne obvezno opcijo `date`, če je ne dodamo bo delo zapisano za današnji dan, če jo pa dodamo pa lahko izberemo poljuben datum.

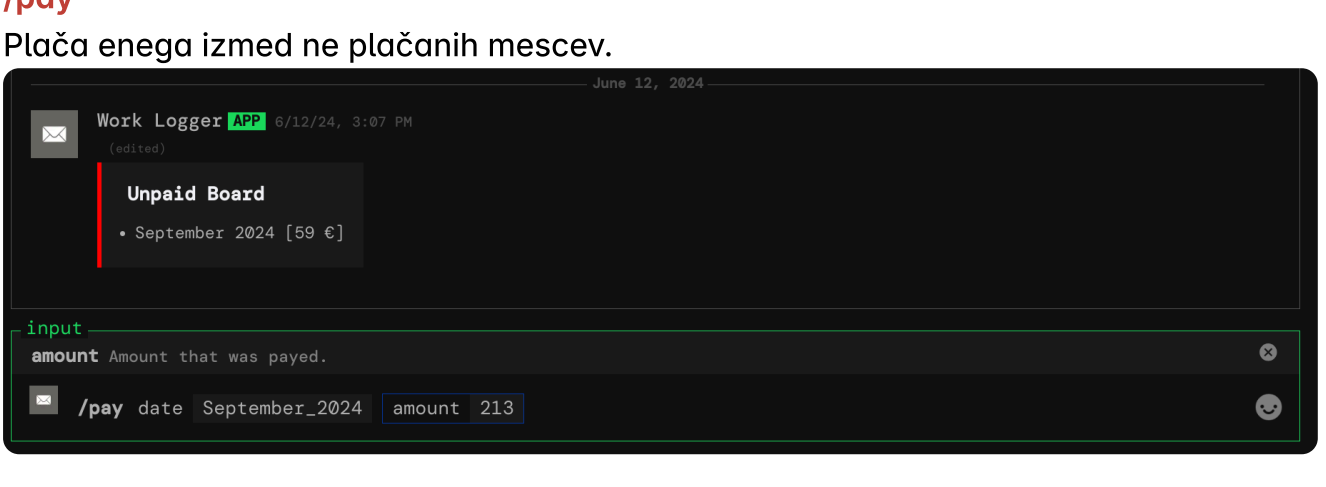




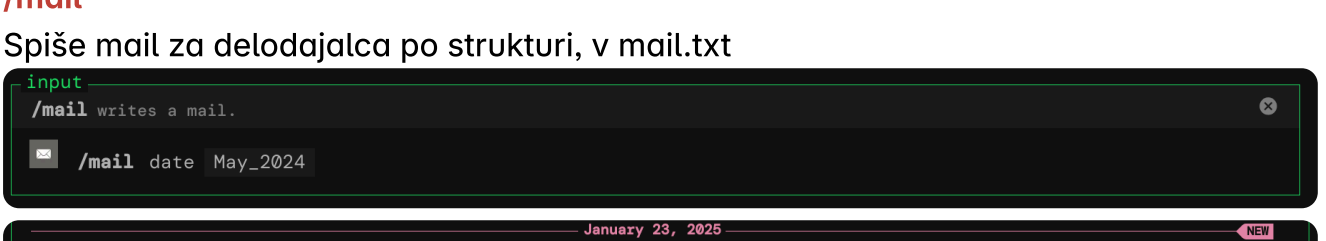
/rm
Remove komanda izbriše en vpis po datumu tipu dela in imenu.
(potencialna izboljšava z podatkovno bazo brisanje pi ID-ju)

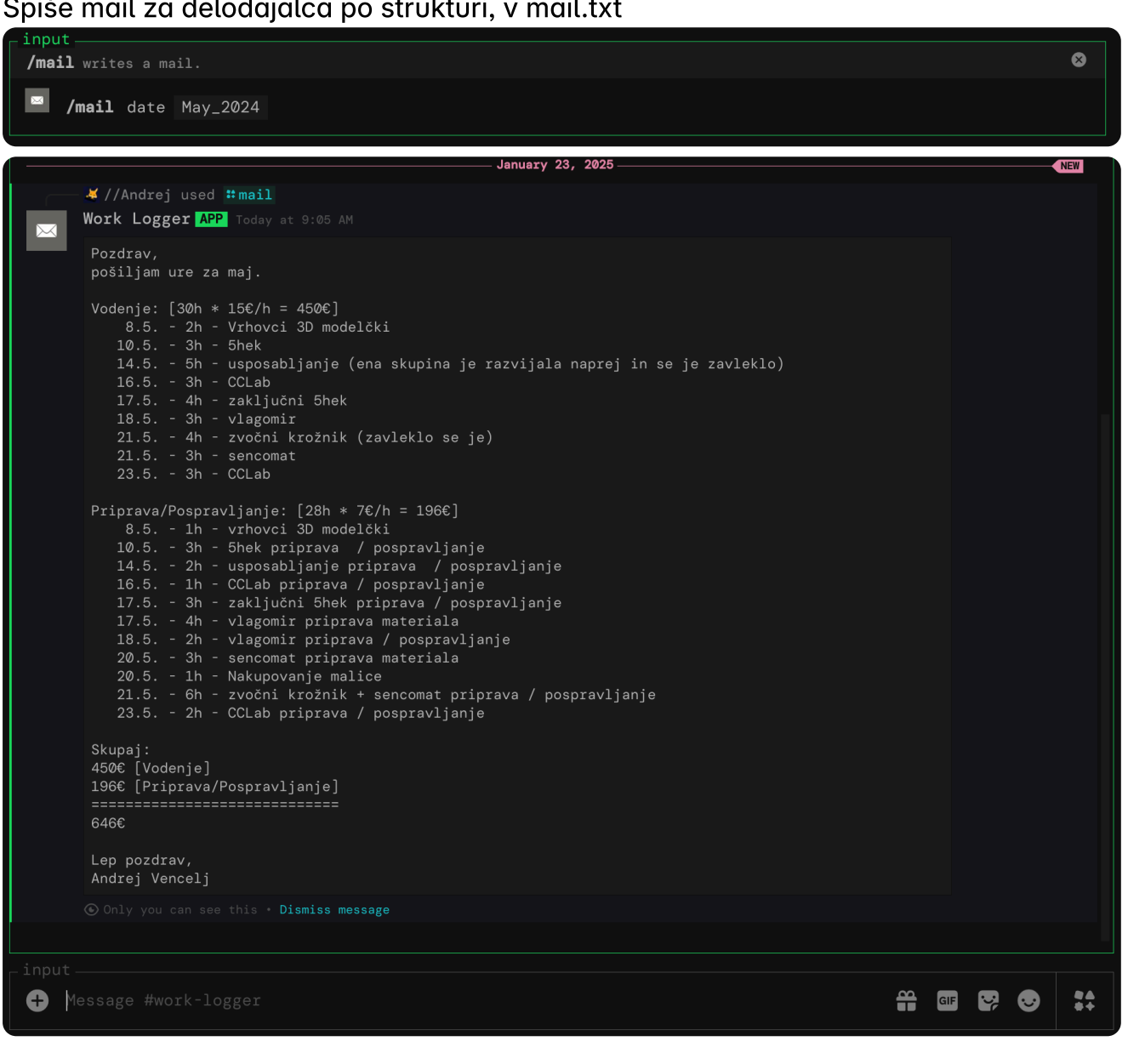


/pay
Plača enega izmed ne plačanih mescev.



/mail
Spiše mail za delodajalca po strukturi, v mail.txt



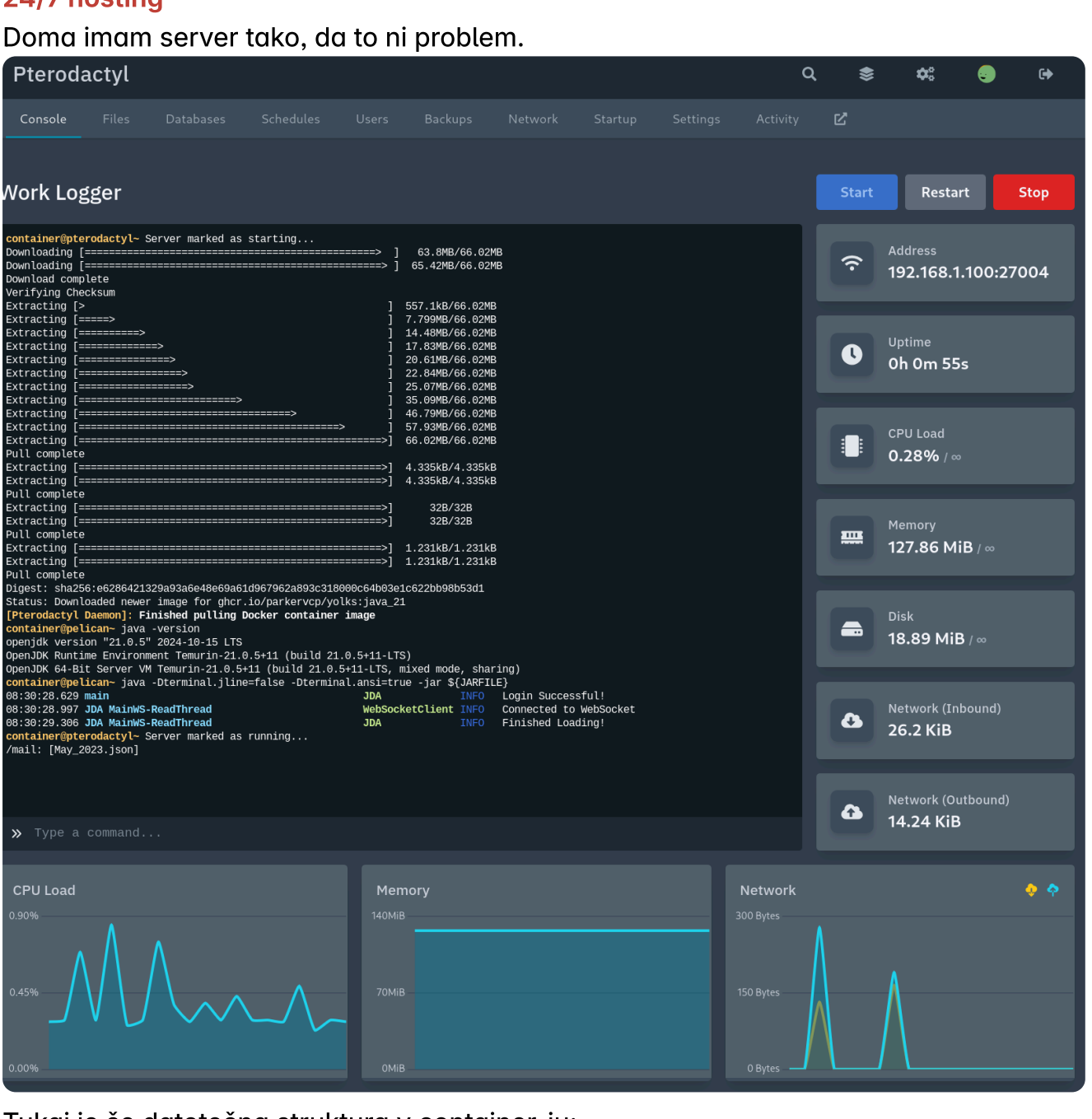


Koda
Vsega ne morem kar pokazati (razen če res žalite), saj je vsega skupaj tam okoli 3000 vrstic. Večina se nahaja v masivni datoteki util.java kjer so definirane vse funkcije (slaba struktura, potencialna izboljšava).

Morda lahko malo razložim main:
Na vrhu imamo ustvaritev config-ov če še ne obstajajo.
Inicializacija/nalaganje config-ov
začetek povezave z api-jem
registracija vseh komand
začetek event poslušalca

```
package io.github.andrejfox.worklogger;

import bablabla ...  veliko importov ...

public class Main {
    public static JDA api;
    public static void main(String[] args) {
        Util.createDefaultIfNotExists("mail.txt", "/default-mail.txt");
        Util.createDefaultIfNotExists("config.toml", "/default-
config.toml");
        Util.loadConfig();
        Util.loadTagOrder();

        api = JDABuilder.createDefault(Util.CONFIG.botToken())
                .enableIntents(
                        GatewayIntent.GUILD_MEMBERS,
                        GatewayIntent.GUILD_MESSAGE_REACTIONS,
                        GatewayIntent.GUILD_MESSAGES)
                .addEventListeners(
                        new AddCommand(),
                        new MailCommand(),
                        new RmCommand(),
                        new PayCommand()
                )
                .build();

        api.updateCommands().addCommands(
                AddCommand.register(),
                MailCommand.register(),
                RmCommand.register(),
                PayCommand.register()
        ).queue();

        api.addEventListener(new ListenerAdapter() {
            @Override
            public void onReady(@NotNull ReadyEvent event) {
                Util.updateNotPayedBoard();
            }
        });

//        Used to add freshly added files to not paid list
//        Util.refreshToNotPayedList();
    }
}
```

24/7 hosting
Doma imam server tako, da to ni problem.



Tukaj je še datotečna struktura v container-ju: