- web4
- express.js za to, da povemo kaj je kašen endpoint
- middleware:
  - koda, ki se izvede, ko dostopamo do nekega endpointa, pred ali po requestu
  - middleware lahko dela neke stvari preden request pride do javascript express strežnika, npr. preverja jwt tokene
  - `res = next()` - pošilji request naprej naslednjemu middlewaru
- imamo neke routerje:
  - kako nej pride zadeva pravilno do končnega requesta - handlerji za posamezni endpoint
  - `/post/new`, `/post/edit` ...
  - imamo npr. post router, ki gre na `/post`, potem imamo še child routerje na `/new`, `/delete` ...
- view:
  - template, podobno kot v flask, samo da uporabljaš ejs
  - z `<%` delamo if, else, for stavke ...
  - ker je to samo template, bomo v brskalniku videli samo končni rezultat, kaj se noter vstavi
- `app.use(cookieParser());` - pobere cookieje iz requesta
- `morgan` - logger
- `jwt_middleware`

Tradicionalni session:

- server ima bazo (oz. redis cache bazo)
- ko se client logina, se v cache bazo shrani session s session id (sid), username, email ...
- kaj če imamo toliko userjev, da imamo veliko serverjev po celem svetu - rabimo gledati po več bazah in delamo dodaten latency
- JWT - vse podatke o seji ima user, tako da ne rabimo cache baze
- payloada in headerja ne moremo spreminjati, ker bi se tako spremenil signature
- moraš samo preveriti, če je JWT veljaven s seeerverjevim private keyem
- da dobimo public key gremo na `localhost:8080/badjwks`
- s public key preverimo, da je naš token res zgeneriral tisti server
- kaj če damo samo v tokenu "sub": 1 in "username":"admin" => spodaj signature ne bo valid
- JWT ne vzame končnih "=" iz base64

- pri nas je problem, da sploh ne pregledamo signatura, ampak samo rečemo `jwt.jwtDecode(token)`, kar samo decoda in ne pregleda signatura
- rešitev:
  - ``const jwt = requite("jsonwebtoken");
  - `req.user = jwt.verify(token, config.PUBLIC_KEY, {algorithms: config.JWT_ALGORITHMS,});` ...
- imamo asimetrične in simetrične algoritme:
  - tukaj dovolimo RS256 in HS256 - ko zamenjamo algoritem, lahko z javnim ključem podpisujemo vse
  - JWT public key mismatch vulnerability
  - jwt_tool.py python knjižnica (https://github.com/ticarpi/jwt_tool)
  - jwt_tool -X -k
- rešitev:
  - dovolimo samo asimetrične ali pa samo simetrične ključe
  - pri nas bi lahko tudi dovolili samo RS256, ker nič drugega ne uporabljamo za signanje
- JWT ima tudi None algoritem:
  - to je bilo mišljeno, da bi bilo uporabno za interne service, ker v interni mreži vsem zaupamo, ampak hočemo nek standarden setup za authentication, da ne rabimo svoje zadeve razvijati
  - naivne implementacije imajo by default vklopljene vse algoritme - med temi tudi None