Django FullStack osaamisesta

Tämä dokumentti on kuvaus Django Framework sovellusohjelmiston ympärille rakentuvasta osaamisestani. Se tukee muuta osaamiseni esittelyä.
Dokumentti on samalla kuvaus osaamisenkehittämishankkeesta. Se kuvaa teknisen kehitysympäristön ja laaditut ohjelmistot projekteina.

Alunperin suunnitellut kehityshankeet, joita on kahdeksan kappaletta, lähestyvät Djangoa aina hieman eri suunnalta. Tämä on mahdollistanut hyvin laaja-alaisen oppimisen ja myös luonut otollisen ympäristön syntyä lisää “sivuprojekteja”, joissa on paneuduttu jo opitun pohjalta eteen tulleisiin uusiin haasteisiin. Nämä “SpinOff projektit” ovat esimerkiksi tutustumista muihin tuotteisiin, kuten frontend kirjasto Vue.js tai grafiikkakirjasto D3.

Sisällysluettelo

Kehitysympäristö....2
Projektit, see also https://github.com/jtapiovaara....2
- P1 CvApp....2
- P2 R4C3....3
- P2 aatooseetrii, uusi versio....4
- P3 Book....4
- P4 Runoja....5
- P5 Blogi....6
- P6 Apishots....6
- P7 Luonto....7
- P8 Kelitiedot....8

Erilliset ja ‘spin-off’ projektit....9
- P12Lmm....9
- P20NatureSites....9
- P21DjongoToMongo....9
- KaroArt....9
- TaLkv....9
- Kia Communique....10
- MSU....10
- Todo....11
- National Parks....11
- Subscriptions....11
- JustyForms....11
- Kauppapaikka....11

JavaScript Projektit....12
- React.js....12
- Vue.js....12
  - Vue-js-client-crud....13
  - Supercool....13
  - Take-all-apis....13
  - Djangovue....13

Thonny....15
- Jutellaan....15
- KiaStocks....15

Kolmoset.....15
P17MunChatti.....15
Django REST Framework.....16
Teknologiat ja työkalut.....17
About Webserver.....17

Kehitysympäristö

Sovelluskehitys tapahtuu PyCharm Professional versiolla 2019.3 (=>2020.23). Koneena on 64 -bit Asus, jossa i5-8250 prosessori ja 8 GB muistia. Käyttöjärjestelmänä on Ubuntu 18.04.3 LTS (=> 20.04.1 LTS). Python on versiossa 3.6 (=>3.8) ja Django versiossa 3+.
PyCharmissa käytetään Git versionhallintaa. Osa projekteista on myös GitHubissa (myöhemmin kaikki loppuun saatetut). GitHubin kautta myös jaetaan valmiita ja vähemmän valmiita projekteja osin aktiivisestikin, muttei alati. Kehityksen tuotantoympäristö on mainittu Asus-läppäri, “koti-WLAN”, jossa on kiinni usean käyttäjän Android/iPhone älypuhelimet ja kaksi erikokoista tabletia (iPad). Lisäksi p2todo on siirretty pythonanywhere.com pilvipalveluun (jtapiovaara.eu => aatooseetrii.eu => www.jtamefiles.fi/projektit ja www.karoart.fi/taide) ja p3book siiretty Herokulle (https://frozen-waters-29964.herokuapp.com/books/).

Projektit, see also https://github.com/jtapiovaara

Hanke koostuu kahdeksasta projektista. Kukin pyrkii lähestymään Djangoa hieman eri suunnilta täydentäen toisiaan ja mahdollistaen opitun käyttämisen aina uudella tavalla. Päätös “Miksi kahdeksan” on tarkoituksellinen. Se ohjaa ajattelemaan tätä työtä yhtenä hankkeena, joka muodostuu valmiiksi saatettavista osista. Lisänä syntyneet muut projektit muistuttavat kaiken projektityön etukäteen suunnittelun vaikeudesta: 13 SpinOff projektia.

P1 CvApp

esittelee Django -projektit. Projektin tarkoitus on toimia omassa wlanissa projektien esittely-ohjelmana tukemassa osaamisen esittelyä. Kustakin projektista on tarkemmin teknisesti ja toiminnallisesti omina sivuinaan. Projektit ovat: p2todo, p3book, p4runoja, p5blogi, p6apishots, p7luonto ja p8kelitiedot. SpinOff projektit ovat oina esittelyinään ja isäksi esitellään laajemmin osaamista Menu -valikossa.
P1 CvApp on puhdas standalone Django -toteutus SQLite -tietokannalla.

Varhainen P1CVAPP

P1 CvAppsta on myös varmuuskopioitu versio omana projektikansionaan “p1cv_BU_20200206”, joka oli ensimmäinen laatimani Django -kokonaisuus. Ei yhtä ‘slik’ toteutus, sisältäen silti seuraavat toiminnallisuudet:

- **Jtaappit** on projektin pääsivu ja tarjoaa kullekin osasovellukselle oman yhtenäisen aloitussivunsa. Sivu on “home” -sivu, jolle voi palata kaikista sovelluksista yhteisen menu -valikon kautta.
- **Tapsa** on sivusto, johon voi ajatella verkkokaupan jatkoa. Täältä löytyy (näyttävä) esittely omaan kirjastooni keräämistä isäni kuvittamista kirjoista. Sovellus esittelee kuvien kautta tapahtuvaa sovelluksen käyttöä (eventit kuvissa, mm. hoover).
- **Luonnonmuistomerkit** on ollut tietojen keruun ja tutkimisen kohteena minulla jo vuosia. App kykenee esittelemään ja hallinnoimaan laaja-alaisesti tietoja kaikista maamme luonnonmuistomerkeistä. App onkin koko sovelluskehitysryppään ainoa ”oikeasti” arvokas kokonaisuus: vastaavaa viranomais- tai muutakaan rekisteriä ei ole missään muualla. Tietosisällöltään tämä on siis ainutlaatuinen. Kohderivejä on noin 4500. Sovelluksen avulla voi esitellä SQLite -tietokannan kykyä ja nopeutta. App voi toimia hyvin esittelyvälineenä itse kohteisiin. Kohteiden esittelyyn on pyritty tuomaan näyttävyyttä Bootstrap -osien avulla sekä lisäämällä hyviä linkkejä lisätietoihin ja myös jakamiseen (Facebook, eMail, WhatsApp, Twitter, ‘copy link’).
- **Traincrud** on rekisteri luetuista kirjoista. Aloitin sen CRUD -toimintojen oppisen edistämiseen ja esittelyyn. App pyrkii käyttämään toteutuksessa mahdollisimman standardeja python/django -ominaisuuksia, kuten *geneerisiä* näkymiä (view), tietokantapohjaisia käyttäjän lomakkeita (form) ja valmiita verkkosivujen pohjia (template).
- **Galleriakierros** on linkki sosiaaliseen mediaan (Instagram). Tätä palvelua ei ole tarkoitus viedä esittelyvaihetta pidemmälle tässä vaiheessa. Liiketoimintamahdollisuutta voi esitellä.
- **Ploki** toimii (staattisena) päiväkirjana erilaisten Django -ominaisuuksien opettelussa. Blogi on omana kehitettävänä projektinaan (P5).

1. Generic View List and Detail.
2. Edustava ovi CVhen, kuvia, flatpages, hoover.

P2 R4C3

on mielestäni tyylikäs Single Page Application (SPA) -kokonaisuus, johon on yhdistetty eri koulutusten lopputulosta. Projekti oli aluksi *tehtävälista*, helppo ja toimiva. Tämän toiminnallisuuden vuoksi päätin tehdä tästä ensimmäisen “deploymentin” Pythonanywhere.com -palveluun. Projekti kuitenkin eli, lähinnä käyttäjäpalautteen vuoksi, ja itse tehtävälista -toiminnallisuus jäi (on toki varmuuskopioituna). Sivuun laitettuun osaan jäivät käyttäjäkohtainen tehtävälista, käyttäjän rekisteröityminen sekä “Seuraajan” rekisteröinti, jossa lähetetään sähköpostia sovelluksen kautta. Sähköpostin osoitteiston suojaamisessa ympäristömuuttujat ovat omassa (suojatussa) tiedostossaan. Lisänä oli myös “Päivän kysymys/Kyselytukimus” App, joka toimi esimerkkinä nopeasti tuotettavasta vaihtelevasta sisällöstä.

Nyt P2ssa on karu Covid-19 laskuri, OpenWeather -palvelun Sää-widget, kaksi muuta itse laadittua säähän liittyvää API -hakua, satunnaisluvulla operoiva “Kissafakta”, Yhdysvaltalaisia luonnonpuistoja esittelevä linkitetty “carousel”, viisaita lausahduksia niinikään satunnaisfunktiolla hakeva “wisewords” ja lopuksi nopea “riisuttu” Google -haku.

Tämän projektin tarkoitus on palvella ryynäämisalustana opetellessa helpompaa suoraviivaista sovelluksen käyttöönottoa. Tavoite on olla nimenomaan ohjeistus ja kokeilukenttä sovellusten käyttöönotolle, eikä tarkoitus ole olla paras esitys kaikesta tehdystä. Mutta siihen voi lisätä kiinnostavimmat osasovellukset (App). Ensimmäinen onnistunut Deployment oli 21.01.2020. ja se tehtiin pythonanywhere.com -sivuston pilvipalveluun. Lisätty 05.02.2020. sää-widget -scripti, joka hakee OpenWeatherMap -sivustolta löytyvän Helsingin kaupungin säätilan. Erillinen laajempi P8 Kelitiedot on omana projektinaan. Lisätty myös satunnaisluvulla operoiva “Kissafakta”.
Settings.py:

```
CRISPY_TEMPLATE_PACK = 'bootstrap4'
LOGIN_REDIRECT_URL = 'home'
LOGOUT_REDIRECT_URL = 'home'

from .email_info import EMAIL_HOST, EMAIL_USE_TLS,\

    EMAIL_USE_SSL, EMAIL_PORT, EMAIL_HOST_USER, EMAIL_HOST_PASSWORD, DEFAULT_FROM_MAIL
EMAIL_HOST = EMAIL_HOST
EMAIL_USE_TLS = EMAIL_USE_TLS
EMAIL_USE_SSL = EMAIL_USE_SSL
EMAIL_PORT = EMAIL_PORT
EMAIL_HOST_USER = EMAIL_HOST_USER
EMAIL_HOST_PASSWORD = EMAIL_HOST_PASSWORD
DEFAULT_FROM_MAIL = DEFAULT_FROM_MAIL
```

P2 aatooseetrii, uusi versio

P2 siirtyi lepoon verkosta ja laitoin paikalle tyylikkään vedoksen. Single Page Application (SPA) -periaatteella toimiva kompakti palvelu, jossa erittäin kovaa ja osin hauskaakin dataa.

Paljon hakuja avoimista rajapinnoista, json.tiedostoista ja omista SQL-kannoista. Hieno kuvakavalkadi karusellina (“swipe if you like”). Mukana myös avoin REST ful rajapinta mainittuihin National Parks -kuviin.

1. aaseetootrii.eu.pythonanywhere.com.
2. Single Page Application.
3. Covid-19 seuranta rajapintahaku Amazon S3sta => rerouted 04/2020 to John Hopkins University.
4. SääWidget API, lisäksi kaksi erillistä (ennuste)hakua rajapinnasta.
5. Kissa fakta, SQLite random -haku.
6. National Parks, linkki ao. Sivustolle. Myös Api: aaseetootrii.eu.pythonanywhere.com/aaseetootrii/api.
7. Wise words, paikallinen json, random -haku.
8. Google.

P2 tilalla pythonanywheressä on tällä hetkellä kaksi omaa sovellustani, P1CVAPP ja aatooseetriin paikan ottanut KaroArt -taitelijan taulukauppa.

P3 Book

Tietokannan hallinta- ja hakutoimintoihin keskittyvä muutoin selkeä ja suoraviivainen projekti. Projekti on ensimmäinen “ihan oikea” aidosti pilvipalveluissa toimiva App (Heroku - PostgreSQL -

S3). Sovelluksen tietomalli on laadittu sisältämään One-to-Many ja Many-to-Many suhteita taulujen välillä Djangon kirjastojen opettelua silmällä pitäen. Projektissa on vain yksi App.

Projektin tietokanta on siirretty PostgreSQL kantaan (Tukholmaan) 2020-01-30.
Projektin kuvat on siirretty Amazon WebServices S3 -palveluun (Yhdysvallat) 2020-02-12.
Projektin palvelin on siirretty Heroku -palveluun (Yhdysvallat) 2020-02-16.
Kehitysversiossa “p3book – development” on joukko etukäteen suunniteltuja muutoksia kuten kuvien koot ja erillinen blurrattu kansikuva. Suunnitelluista muutoksista on projektidokumentaatiossa (Great Ideas – cunning plans). Tätä ei ole tarkoituskaan viedä ‘tuotantoon’, mikä näin palvelee koko hankkeen tarkoitusta (=> Projektinhallinnan osaaminen).

Settings.py:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'cekkztgc',
        'HOST': 'balarama.db.elephantsql.com',
        'USER': 'cekkztgc',
        'PASSWORD': 'MUC9Juuuta6cXfonyveE-y2i9oklkaXm',
        'PORT': '5432'
    }
}

AWS_STORAGE_BUCKET_NAME = AWS_STORAGE_BUCKET_NAME

AWS_S3_REGION_NAME = AWS_S3_REGION_NAME



ALLOWED_HOSTS = ['frozen-waters-29964.herokuapp.com']
```

1. Kolme taulua ristiinviitattuna.
2. PostgreSQL, ElephantSQL, Heroku, AWS S3.
3. Kehitysverisio ja tuotantoversio eriytetty (Git).
4. Projektinhallinta kurinalaista.
5. frozen-waters-29964.herokuapp.com.

P4 Runoja

Lähtökohtana oli aktivoida käyttäjää kirjoittamaan runoja ja sen jälkeen palaamaan palveluun hakemaan ylläpidon/backofficen antamaa palautetta. Lyhyiden runojen kirjoittaminen (tai lausuminen!) arvioidaan erikseen tallennuksen jälkeen. Tähän voi ajatella sosiaalisen median liittymää tykkäyksineen ja 1-5 -arviointeineen. Käyttäjä palkitaan modernia uussanaa esittävällä tms. kuvalla. Valitun tai kaikki runot voi myös tulostaa pdf-muotoon. Appien uudelleen käyttöä edustaa sovellukseen tuleva “seuraaja” -toiminto ja käyttäjän rekisteröintitoiminto (alunperin P2todo -projektista). Käyttöliittymässä on dynaaminen Carousel -luuppi. Teknisenä tavoitteena oli luoda CRUD -toiminnot, joka toteutuikin. Projektia on käytetty myös BackEnd -harjoittelussa ja siinä on (tarkoituksella) joukko “ylimääräisiä” tauluja (models.py). Käytetty pohjana kahdessa SpinOff -projektissa: KaroArt ja TaLkv.
P5 Runoja on puhdas standalone Django -toteutus SQLite -tietokannalla.

Erityistä. Tulostuksessa käytettävä Weasyprint vaatii hieman erityishuomiota. Alla toimiva esimerkki, jossa on alleviivattuna vaadittu lisävääntö, jotta kuvat tulevat mukaan:

```
def html_to_pdf_view(request, *args):
      “”” this prints an eBook “””
      paragraphs = Post.objects.filter(*args)
      html_string = render_to_string('pdf_template.html', {'paragraphs': paragraphs})
      html = HTML(string=html_string, base_url=request.build_absolute_uri())
      html.write_pdf(target='/tmp/eKirja.pdf')
      fs = FileSystemStorage('/tmp')
      with fs.open('eKirja.pdf') as pdf:
            response = HttpResponse(pdf, content_type='application/pdf')
            response['Content-Disposition'] = 'attachment; filename="eKirja.pdf"'
            return response
      return response
```

1. CRUD, jossa myös käyttörajoitus implementoitu (delete).
2. Käyttäjän input + palkitsemisen myötä paluu.
3. Output. Mahdollisuus tulostaa runo/runot pdf-muotoon.
4. SpinOffs.

P5 Blogi

Siirsin blogin omaksi projektikseen. Katson, että kun se on omana kokonaisuutenaan, niin tarina pääsee paremmin esille. Toiminnallinen muutos osaamisessa ja kiinnostuksen kohteissa tulee esille. P1CvApp -projektiin “ploki” jää palvelemaan CV:n tarpeita: varhainen opiskelun aloittaminen ja työn edistyminen.

Lyhyet mutta usein toistuvat päivitykset kuvineen sallivat myös käyttäjien kommentoinnin. Blogista on syntynyt SpinOff “KiaCommunique”. Lisätty 03/2020 D3 -grafiikkakytkin, jolla näytetään tehtyjen blogien määrä per kuukausi. Vähän, mutta tavoitellen luonnollista modernia blogi -lookoutia.
Koko blogin voi tulostaa eKirjaksi halutessaan.

1. Ylläpito, bloggaus, toteutettu Django Adminia hyödyntäen.
2. ‘Matkakertomus’, tukee CVtä.
3. D3 -grafiikkaa.

P6 Apishots

Django -kehittämisen ja avoimen koodin yleensä iso ajatus on, noh, avoimuus. Tämä projekti käy läpi Django REST Frameworkin avulla API tapahtumia. Projektissa on vain yksi App, ohjelmoijia ja heidän osaamistaan esittelevä“apis”, jossa on kolme taulua. Opitaan REST, JSON ja myös API -testaus/hallintatyökalu Postman. Web Tokens (JWT) pintaa sipaisevat alkeet. App tunnistaa myös “oikean” tavan käyttää API -rajaa siten, että tunnistautuminen ja muutokset tapahtuu muualta käsin

kuin sovelluksen sivuilta (esim. mainittu Postman.). Todo -projektissa (SpinOff) lisäksi React.js. Ja Vue.js FullStack -toteutukset.
Settings.py:

```
# Hallitaan käyttäjien pääsyä crud APIssa näkyviä tietoja
#
# permissions.IsAuthenticatedOrReadOnly - voi lukea, muuttei muuttaa. Login voi crud
# permissions.AllowAny  -  Kuka vaan voi crud.
# permissions.IsAuthenticated , Vapauta jälkimmäinen luokka alla - Vain kirjautunut voi crud. Tällöin pitää olla
# Token esim. Postmanin kautta tai kts. PrettyPrinted video JSON Web Tokens With Django REST Framework
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': ('rest_framework.permissions.IsAuthenticated',),
    'DEFAULT_AUTHENTICATION_CLASSES':
('rest_framework_simplejwt.authentication.JWTAuthentication',)
}
```

1. Django – RESTful API.
2. Postman -työkalu.
3. JWT Tokens.
4. React.js ja Vue.js SpinOffs.

P7 Luonto

Luonnonmuistomerkit omana projektinaan. Projekti palvelee kahta tarkoitusta: tuoda jo olemassa oleva Python -sovellus sisään Djangoon sekä hallita suuren (20k json-rivejä) tietomäärän lataaminen. Projekti on combo kolmesta eri suunnilta asiaa lähestyneestä appista:

- kuvattiin mahdollisimman monisäikeinen models.py
- rakennettiin REST Api ja ladattiin reilut neljä tuhatta kohdetta kantaan
- luotiin “fullstack” ajattelulla kaikki tarvittava tuotantokäyttöön

Projektista on kolme (teknistä) SpinOffia: P12, P20 ja P21. Näistä P12 toimii kehitysversiona (2020-04 =>).
P7 käyttää Djangon geneerisiä malleja (Views, Forms) mahdollisimman laajasti. Kohteiden kuvat (models.Photo) ovat AWS S3 -pilvessä. Ympäristömuuttujat omassa (suojatussa) tiedostossaan. Kuvista on toteutettu myös API. Suurta joukkoa rivejä tukee käyttöliittymässä pagination. Sovelluksessa on toteutettu myös front-end osio (React.js ja Vue.js).
Settings.py:

```
INSTALLED_APPS = [
    '...',
    'storages']

from .photo_info import AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, DEFAULT_FILE_STORAGE,\

    AWS_STORAGE_BUCKET_NAME, AWS_S3_REGION_NAME
AWS_ACCESS_KEY_ID = AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY = AWS_SECRET_ACCESS_KEY
DEFAULT_FILE_STORAGE = DEFAULT_FILE_STORAGE
AWS_STORAGE_BUCKET_NAME = AWS_STORAGE_BUCKET_NAME
AWS_S3_REGION_NAME = AWS_S3_REGION_NAME
AWS_BUCKET_ACL = None
AWS_DEFAULT_ACL = None
```

1. Paljon dataa (+ 4k riviä)
2. Etsi -toiminto (Filters.py)
3. Kuvat AWS S3ssa
4. Testattu Heroku -deploymentia varten
5. Front-End harjoitukset (React.js)

P8 Kelitiedot

Helppo tapa seurata säätiloja maailmanlaajuisesti. Sovellutuksen oppi on helppo säädatan käyttö avoimesta OpenWeatherMap -palvelusta. Erityisesti haetun datan parsimista (JSON) ja yksinkertaisen nopeasti tuotetun sovellutuksen iso hyöty. Tässä projektissa toteutetaan myös käyttöliittymän valmiin suunnitelman hyödyntäminen (html pohjana weather.html tiedosto, joka poimittiin PrettyPrinted.com palvelun videotarjonnasta. Myöhemmässä P8 versiossa tämä malli on stilisoitu enemmän omanlaisekseni perus- ja Bootstapmalliseksi, mutta kehitystyön ajan se palveli hyvin ja oli erinomaisen opettavainen). Samaa oppia hyödynnettiin myös kahdessa muussakin spinoff -projektissa (KaroArt, TaLkv).

1. SääAPI
2. JSON parsinta
3. Olemassa olevan suunnitelman (html.tiedosto) hyödyntäminen kehityksessä.

Erilliset ja ‘spin-off’ projektit

Vaikka lähtökohtaisesti teenkin kahdeksaa osaamisen kehittämis- ja esittelyprojektia, niin ei vaan voi mitään, että muuta uutta syntyy siinä samalla. Näissä monessa jopa vapaammin ja villimmin hienoja yrityksiä (jos erehdyksiäkin).

P12Lmm

P12 on LMM esittelyversio, johon on kasattu “parhaat” +1200 kohdetta. Kaikista on kuva, päätösdokumentti ja tieto “Tie”. Dashboard tyylinen kohdesivu. Django- SQLite-Charts.js-Vue.js. P12 muotoutuu luonnonmuistomerkkien pääteokseksi. Mukana kaikki Suomen maakuntien ja kuntien vaakunat, joiden mukaan sovelluksessa navigoidaan. List-list-list-detail normaalia haastavampi hierarkirakenne. Kohteessa monipuolisesti tietoja ja teknisesti erilaisia linkkejä eteenpäin.

```
"""Faktakartta yksittäisen kohteen tietoihin. Kahdeksan tietokokonaisuutta kustakin kohteesta.
1. Kunnan päätökset ja tämän ajallinen sijoittuminen niissä.
2. Kaikki oleellinen (self -kentät).
3. Tyyppi -kuva (piirros).
4. Valokuva kohteesta
5. Kartta. GoogleMaps -linkki.
6. Päätösdokumentti. Linkki aukeaa pdf-dokumenttiin.
7. Somefeed.
8. tags.  tietotyypit haku (tyyppi, svuosi, tie)"""
```

P20NatureSites

P20 luonnonmuistomerkkidatan tietokannassa (SQLite) on yli 4000 kohdetta. Toiminnallisuutta voi lisätä suunnitelmallisesti, mutta nykyisetkin toiminnallisuudet ovat jo merkittävän hyvät. Palvelua “odottaa” hallittu deployment, se on tarkoituksellisesti jätetty tähän vaiheeseen.

P21DjongoToMongo

Otsikostakin voi päätellä. Tässä sivu-projektissa toteutettiin suuren määrän luonnonmuistomerkkidataa siirto MongoDB dokumenttikantaan. Tietokanta on GoogleCloudServices pilvipalvelimella ja voi hyötyä suunnittelussa jo aiemmin toteutettua REST rajapintaa.

KaroArt

Carousel, Omat myytävät, Ostajakontaktointi. Yleiskäyttöinen taiteilijan henkilökohtainen teosten esittely- ja kauppapaikka. Palvelu on spin-off p4Runot Appista. Tämä on (nuoren) taiteilijan teosten kauppapaikka. Käyttöönottovalmis ja hyvä kehityssuunnitelma (jota toteutetaan osin p4 -projektissa). Valmis tuotannossa www.karoart.fi/taide.

TaLkv

Carousel, Omat myytävät, Ostajakontaktointi. Kiinteistövälittäjän henkilökohtainen tehtävä- ja kontaktityökalu. Palvelu on spin-off p4Runot Appista. Hyvin mielenkiintoinen mahdollisuus luoda

kiinteistövälittäjän henkilökohtainen kontakti- ja kohdetyökalu. Mm. kohteiden “flip-over”, kohteeseen kohdistaminen kääntää “kortin”, jonka toisella puolella lisätietoja.

Kia Communique

Spin-off omasta blogistani. Uudelleenkäytettävä suunnittelu mahdollistaa kevyen ja itse hallittavan blogin ylläpidon. Toimii jo tässä kehitysvaiheessa hyvin tehtyjen bloggausten ja muiden (julkisten) esitysten alustana.

MSU

Olen kuluneen vajaan kymmenen vuoden ajan tehnyt useita erilaisia osallistumisia **StartUp** -yrityksiin. Pariin omaankin.
Tämä App listaa nämä tekemiset ja on siinä mielessä “CV” työ ja sopii liitettäväksi P1 -projektiin.
Teknisinä erikoisuuksina on monipuolinen QuerySet -käyttö (mm. ‘get_context_data’, ForeignKey ja ManyToManyField tapauksia). Lisäksi Djangon rakenteen ulkopuolisen ‘snippet’ -tiedoston käyttö. Views.py on toteutettu kommentoimalla mahdollisimman tarkkaan ja oppimalla huolella eri tekniikat (Joskus jossain jollain nyky-ohjelmoijalla saattaa olla ollut tendenssi kopioida toimintoja suoraan niitä täysin ymmärtämättä.).
Etuisuuksia esitellään Charts.js -kirjaston kaaviolla. StartUp’s omana menunaan.

Ja vielä: Myös retrohenkinen API -harjoitus Saksan Dusseldorfin alueen **autokaupparekisteristä** (‘https://offenedaten.duesseldorf.de/api/action/datastore/search.json?resource_id=9a0b8848-2369-4c05-94c4-550c5990b7f9‘).
Ja vielä: Visuaalilisä **Pörssi**. Lista tietyistä osakkeista, niiden määrät ja viimeisimmät osingot. Koristeena pörssisijoittamisen kauheutta ja kauneutta kuvaavia animoituja emojeja. Tästä myös henkilökohtaisia sijoituksia hallinnoiva **KiaStocks**, “Spinoff’s Spinoff”.
Ja vielä: Tottakai P2-projektissa luodusta Suomen vallitsevasta **COVID**-19 tilanteesta.
Ja vielä: **Postinumero**. Toteutus aluksi ‘vain’ Django Admin -puolella. Syöttämällä postinumero-palautetaan numeroon liittyvät paikkatiedot. Kaikki työ tehdään suoraan .models -tasolla, siis tietokantatriggeri. Hieno satelliittikuva-lisä.
Ja vielä: **SpaceX** -avaruusyhtiön rajapinnasta haetaan kaikkien rakettien kaikki mahdollinen tieto melko monitasoisesta json-mallista.
Ja vielä: Kiinnostava kokeiltu on myös suoraan views.py tiedostoon Bootstrappeineen kirjoitettu **MunView**.
Ja vielä: Verkkokeskusteluista löydetty CSS-mahdollisuuksia esittelevä **Hehkutuksia**, jossa leikitellään neonväreillä ja linear gradient transparent infinite etc. vivuilla.

```
class MunView(View):
      """Tämä on rakennettu, jotta voin harjoitella ja ymmärtää miten Djangon view -osastot
toimivat.
      Ihan perusview, siis 'view' toimii kokonaan ilman mitään renderöintejä, palauttaen vain
HttpResponsen."""
      def get(self, request):
            return HttpResponse
```

MUS ei ole, näköjään, varsinainen SpinOff, vaan oikeastaan kiinnostuksesta ja itsensä haastamisesta syntynyt kokonaan uusi Python/Django pelikenttäprojekti.

Todo

Pieni app, jolla ensimmäisen kerran on toteutettu tyylikäs Django + React.js -yhteys. Ensimmäinen todellinen FullStack toteutus siis. React.js Appeja tehtiin neljä,kunnes tämä onnistui (React app fourth.js). React -appeista on oma kappaleensa.
Todo pääsi myös mukaan, kun siirryin oppimaan Vue.js -kirjastoa. Toteutin tähän Django-osaan ensimmäisen ehjän Vue FullStack crud -kokonaisuuden. Vue -appeista on oma kappaleensa.
Tämä on koostaan ja toiminnallisuudestaan huolimatta keskeinen ratkaisu koko opiskelussa. Hyvin pienellä Django -komponetilla voidaan yhdistää kaikki mahdolliset ja mahdottomat tietokannat ja myös eri käyttöliittymien ohjelmointiympäristöt. Palvelussa ei ole ollenkaan Djangon Template osaa (paitsi siis REST -rajapinta ja Admin -osa).

National Parks

Nopeaksi API-harjoitukseksi ja näyttäväksi yhden sivun Appiksi yhden aamun (3h) aikana rakennettu palvelu jossa esitellään modernilla ja hauskalla tavalla Yhdysvaltain luonnonpuistoja. Liitetty osaksi P2 -projektia. FullStack toteutus osana React App ‘fourth’-appia ja Vue.js.

Subscriptions

Pieni palvelinsovellus, jossa tallennetaan kotitalouden kuukausitilauksia tyyliin Netflix. Palvelun ydin on REST Api, jota käyttää Vue.js (Djangovue-MyTilaukset).

JustyForms

Rakennettu tukemaan urheiluseurajoukkueen historiaa (tässä oma Ultimate Frisbee -ura). Joukkueen tai pelaajan näkökulmasta tehty rekisteri. Teknisenä kohteena Django Forms + Bootstrap + crispy_forms ja sen kautta tehtävä huolellinen tietojen syöttö. Pelaajien kuvat tallennetaan suoraan Amazon S3een. Yksityiskohtana (vielä), että localhost -käyttö vaatii ao. kuvan kopioimisen paikalliseen ‘pelaajat/’ -kansioon. Puhdas Python/Django CRUD -sovellus omasta Ultimate Frisbee urastani. Voidaan lisätä ja ylläpitää joukkueita ja pelaajia.
Oman pelihistorian voisi tulostaa pdf -muotoon.

Kauppapaikka

Täysin toimiva eCommerce saitti on rakennettu vapaasti seuraten Youtube -kanavaa Django eCommerce Website by Dennis Ivy.
FullStack Web, jossa FrontEnd osia on koodattu JavaScriptillä ilman erityisiä frameworkeja (React, Vue,…). Sovelluksen BackEnd päässä on realistisesti rakennettu tietokanta, jossa mm. customer, order, order item ja postitukset on hallittu omina tietokantaosinaan. Myytävät tuotteet voivat olla fyysisiä (fyysinen lähettäminen) tai vaikkapa digitaalista musiikkia (lataaminen). Käyttäjä voi olla kirjautunut (Django Admin kautta) tai olla vain satunnainen sivustolla kävijä. Kauppapaikassa on toteutettuna kirjautumattoman asiakkaan ostamista tukeva **Cookie** -malli.
Ostokset voi maksaa **PayPal** -linkin kautta.

P17MunChatti

Oma itse tehty chatroom. Websockets ja uusia Django osia, consumers.py ja routing.py.

Jutellaan

Google Translator tehtynä itse. Admin -työkalulla voidaan hallita haluttua määrää käännettävistä kielistä. Käyttää translate==3.5.0 kirjastoa kääntämiseen. Capisco!

KiaStocks

Ei olisi ollut erityinen sovellus, pelkästään osa MSU -pakettia. Kuitenkin Python **pandas** -kirjastoon tutustuminen tämän kautta antoi aiheen omaksi projektikseen.

JavaScript Projektit

Python-Django yhdistelmällä voi tehdä melkein mitä vaan web-sovelluksia. Melkein mitä vaan. JS on ollut jo pitkään standardi FrontEnd -koodauksessa, joten on luonnollisesti tärkeätä oppia tämäkin pää. Valitsin Facebookin Reactin ja Google SpinOff Vuen ympäristöiksi. Angularin jätin pois, koska luulen sen olevan hieman menossa pois suosiosta. Kovaa kasvua odotan Vue.js osalle.

React.js

REST ful vaatii siis toisenkin pään, jotta FullStack toteutuu. ReactAPP “fourth” toimi tämän ensimmäisenä opetteluna. Jo useampia pieniä servlettejä vai mitä ne on tehtynä. Myös nimi “fourth” kertoo yrittämisestä useammin, tein harjoitellessa projektin ensin kolme kertaa (first, second, third). Karu ‘Coronas’ sovellus täälläkin. Käyttää Django BackEndissä *Todo_api* rajapintaa SQLite kannasta. Myös Yhdysvaltain kansallispuistot pythonanywhere.com -palvelusta.

```
import ReactYouTubeExample from './thirdparty/ReactYouTubeExample'
import Metronome from './thirdparty/Metronome'
import Todo from "./posts/todo"
import Coronas from "./posts/coronas"
import NatParks from "./posts/natparks"
import PostList from "./posts/PostList"
```

Vue.js

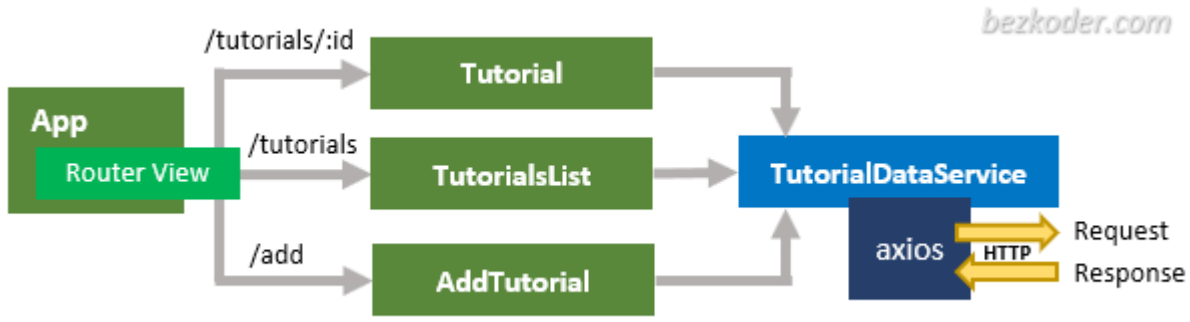
React jälkeen siirrytään uudempaan ja kasvavaan Vue.js frameworkiin. Lähtökohtaisesti avataan ymmärrys Vue -rakenteeseen ja erityisesti REST -rajapinnan käyttö (crud -ratkaisu).
Vuen fiksu päätös yhdistää template-script-style samaan .vue tiedostoon auttaa. Perushakuscripti perustuu standardiin ja eri Vue kokonaisuuksissa suositeltuun muotoon ja näyttää minulla tältä:

```
<script>
    import axios from "axios";
    export default {
        name: "mytilaukset",
    data() {
        return{
            postBody: '',
```

```
                subscriptions: [],
                errors: []
            }
        },
        created() {
            axios.get('http://127.0.0.1:8000/api/subscription_api/')
            .then(response => {
                this.subscriptions = response.data
            })
            .catch(e => {
                this.errors.push(e)
            })
        }
    }
</script>
```

Eri api-rajapinnoista on erikseen kappaleessa Django REST Framework

Kuten kaikissa muissakin projekteissa, tässäkin on korvaamatonta apua saatu eri lähteistä. Vue.js CRUD aukesi lopulta bezkoder.com -sivuston kautta. Toteutetun Django-Vue -FullStackin arkkitehtuurikuva on mainitulta sivustolta.



Toteutuksessa mielitytti erityisesti FrontEndiin tarjottujen palveluiden kasaaminen yhteen samaan tiedostoon (TutorialDataService).

Vue-js-client-crud

Tämä projekti on tehty ylläolevan esimerkin “tutorial” -projektista, joka on muokattu tehtävä -listaukseksi. Toimiva ja hieman jo tyylitelty FullStack CRUD sovellus. Käyttää Django BackEndissä *Todo_api* rajapintaa SQLite kannasta.

Supercool

Tässä projektissa listaus haetaan REST rajapinnasta päivitykseen Vueen. Käyttää Django BackEndissä *Todo_api* rajapintaa SQLite kannasta. Suoraviivaisesti käytetään (ensimmäisen kerran) olemassa olevaa itse tehtyä REST rajapintaa, siitä nimi.

```
components: {
  inventory
}
```

Take-all-apis

Tämä projekti on Django(projekti p21DjongoToMongo)-Vue paketti, joka käyttää Mongo Dbtä. Linkki on siis Djangon kautta, ei suoraan vue-mongo.

```
components: {
   lmmmongo
}
```

Djangovue

Projektissa yhdistän tehtyjä Vue.js appeja hieman hallitummin ja kasaan App.vue -tiedostoon kaikki komponetit. Kukin suunniteltu siten, että voisi olla oma sovelluksensa.

```
components: {
      MyHoveroi - js kokeiluja hoover toiminnoilla
      MyKalenteri - js kalenteri
      MyLmms – listaa p12 projektin luonnonmuistomerkkejä
      MyEkavue - tuo pythonanywhere.com APIn tiedot (National Parks) listalle
      ExampleComponent – tuo jsonplaceholder -palvelun esimerkit listalle
      HelloWorld – helloworld :)
       MyTilaukset – listaa kotitalouden kuukausikohtaisia viihdetilauksia
}
```

Komponenteista myLmm tuo p12 luonnonmuistomerkit listalle. Pythonanywhere.comissa ylläpidossa olevaa Yhdysvaltain luonnonpuistot -palvelua voi tutkia MyEkavuen kautta.

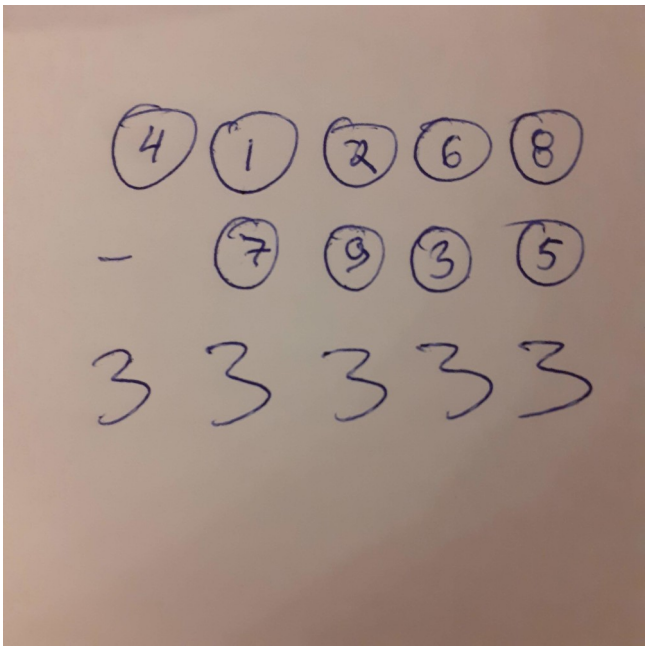
Twotter

Vielä työn alla (20201112) oleva Twitter klooni. FrontEnd puhtaastiVue.js.

Thonny

Kolmoset

Thonnyn kanssa tehtynä Kolmoset -setti. Kolmoset lähti manuaalisesta matematiikkaharjoituksesta. Päätin tehdä saamani matematiikka vähennysyhtälö -tehtävän sovelluksena ja toteutin sen Ubuntun Thonny IDEllä ja puhtaalla Pythonilla. Käyttöliittymään lopuksi kevyesti (ja ei kovinkaan viimeistellysti) Djangoa. Toiminnallisuus oli tässä SpinOffissa tärkein. CV osoitus luovasta ongelmanratkaisusta. Sovellus laskee listalta (1-9) mahdolliset viisi- ja nelinumeroiset luvut , jotka toisistaan vähentämällä tuottavat tuloseksi luvun 33333. Lisäsin myös mahdollisuuden valita minkä vaan luvun lukujoukon sallimissa rajoissa. Ei helppo. Hyvä.



Django REST Framework

Osaan, itse asiassa kasvavassa määrin lähes kaikkiin projekteista on toteutettu API-rajapinta .js-frameworkeja varten.

National Parks: aatooseetrii.eu.pythonanywhere.com/aatooseetrii/api (online)
Todo: localhost:8000/todo_api/ (käytössä Django, React ja Vue)
Luonnonmuistomerkit: localhost:8000/api/spot/
p12LmmApi: localhost:8000/peli/peli
MongoDB palveluun: p21DjongoToMongo/naturesites
subscriptions: localhost:8000/api/subscriptions_api
oma blogini: p5blogi/ploki
P6 -projektissa REST rajapinnat useampaan tauluun suoraan: p6apishots/apis

Lisäksi käytössä on kasvava joukko API-rajapintoja, joista luetaan tietoa REST standardin mukaisesti.

CatFacts, WiseWords, SpaceX, Säätiedot, Saksan Autokauppiaat, … määrä ei ole tässä tärkeä, vaan se, että erilaisia JSON -muotoisia tietoja on haettu lukuisalla eri tavalla.

Teknologiat ja työkalut

About Webserver

Lähtökohtaisesti tässä kehitysympäristössä ajetut sovellukset käyttävät porttia 8000 (127.0.0.1:8000).

Koska palvelimia voi olla useampi yhtäaikaa “pystyssä”, on mahdollista ajaa vaikka kaikkia tehtyjä sovelluksia samaan aikaan. Webserver ideahan on, että palvelin kuuntelee (Listen) tulevia kutsuja ja vastaa niihin pyydettäessä. Näin ollen useampi palvelin samassa kehityskoneessa ei sen kummemmin tee hommaa hitaammaksi tai raskaammaksi.

Tein (20201008) muutoksen varsinaisiin projekteihini kehitysympäristössä. Kukin omalla palvelin tunnuksellaan.

Sovellus	Palvelimen portti	“kännykkä” portti
P1	8001	8801
P2	8002	8802
P3	8003	8803
P4	8004	8804
P5 *	8005	8805

P6	8006	8808
P7	8007	8807
P8	8008	8808
Eräitä Spin-offeja		
P12LMM	8012	8812
MunChatti	8017	8817
MyStartUps		8825
NatParks *		8832
Kauppapaikka		8852
Capisco! *		8862
KiaStocks *		8866

Testasin kaikkia soveluksia käytössä yhtäaikaa.
* -merkityt ovat siirretty “pysyvälle”palvelimelle toiseen koneeseen, vanhaan Dell läppäriin (näille vanhoille jo kauan hylätyille koneille on joka taloudessa käyttöä). Nämä sovellukset pidän auki “koko ajan”, ja ne palvelevat esimerkistä lähiverkon, (koti)WLANin arkkitehtuurista.

Next step on avata palvelu laajemmin verkkoon. “Normaali” Deployment tapahtuu luonnollisesti hallitusti reittiä Git – Heroku/Pythonanywhere. On kuitenkin mahdollista testata ja esitellä sovelluksia myös suoraan kehityskoneelta käsin. Vaikka mobiilisti junamatkalla. Tätä voi tehdä esim. Ngrok -palvelun avulla: “*Ngrok exposes local servers behind NATs and firewalls to the public internet over secure tunnels.*” - ./ngrok http 8000 -.