Haladási napló -

couchsurfing alkalmazás fejlesztése iOS platformon

**Szeptember 9-15.**

* Létrehoztam a GitHub repo-t, amelyben majd láthatóak lesznek a kódok. A repo linkje: <https://github.com/belbebe/couchsurfing_ios>
* Alkalmazásfejlesztői tapasztalat híján elkezdtem egy online kurzust, amely iOS fejlesztéssel foglalkozik
* A justinmind program segítségével elkezdtem az alkalmazás képernyő képeinek a megtervezését
* Létrehoztam egy nem végleges kliens oldali specifikációt, amelyet tovább szeretnék részletezni, illetve szükség esetén módosítok rajta, ez elérhető a következő linken:

<https://docs.google.com/document/d/1MG6L1ZPzAI2ftvTS2guLQ37nQHWmXCmHf22piukXJcw/edit?usp=sharing>

**Szeptember 16-22.**

* Átgondoltam az adatmodellt, és a csütörtöki konzultációs időpontig el is készítem a sematikus ábrázolását
* Részletesebben átgondoltam a kliens oldali specifikációt, ezt is frissítem majd a megfelelő dokumentumban - frissítés megtörtént
* Folytattam az online iOS app fejlesztésről szóló kurzust
* VIPER architektúrával való megismerkedés

**Október 28.**

* Megnéztem a REST API-kkal kapcsolatos videókat a kapcsolódó előadásokról (általános REST API-s videó, Swagger-rel kapcsolatos videó)

**Március 9.**

* View elkészítésének elkezdése
* Ütemterv jelen állás szerint:
  + UI elkészítése, adott esetben bizonyos funkciók implementálása: 5-6. hét
  + Backend elkészítése: 6. héttől 3-4 hét, lehetőleg minél hamarabb

**Március 16.**

* Foglalási oldal majdnem teljesen kész, egyéni igények jelzésével probléma (hogyan lehet kicsi multiselect list-et létrehozni), keresés gomb létrehozása még nem történt meg
* Működéshez szükséges „@Static” tagváltozók (property-k) létrehozása, ezekben a változókban történik az olyan adatok eltárolása perzisztens módon, mint például a textfield-ekből bevitt szövegek („source of truth”). Ahogy a user módosítja az ezekhez a változókhoz tartozó binding változók értékét, a @Static-kel ellátott változók értéke is dinamikusan változik, majd ezen változók alapján történik a View rebuild-elése.
* A Foglalási View összerakása során igény szerint alkalmaztam Vstack-eket és Hstack-eket, ezekben helyeztem el a különböző UI komponenseket, amelyekből létrehoztam a UI-t. Az egyes komponensekhez modifier-eket is hozzáadtam, ezzel létrehozva egy többé-kevésbé egységes kinézetet. A stack-eknél alkalmaztam a „spacing” paramétert, hogy az egyes komponensek egymástól elkülönítve jelenjenek meg.
* A felhasznált komponensek között user inputot lehetővé tevő komponensek is megtalálhatóak, pl. TextField komponensek, illetve DatePicker komponensek. A fizetési módot egy Picker komponenssel (<https://stackoverflow.com/questions/56513339/is-there-a-way-to-create-a-dropdown-menu-button-in-swiftui>) valósítottam meg, amelyben megtalálhatóak a fizetési lehetőségek, a user ezekből tud majd igény szerint választani.
* Az ár kategória meghatározásánál a TextField-ek inputjait szűröm, hogy csak numerikus értékek legyenek eltárolva a megfelelő @Static változókban. Ehhez működőképes megoldást a <https://stackoverflow.com/questions/58733003/swiftui-how-to-create-textfield-that-only-accepts-numbers> linken találtam.
  + Itt meg kell valósítani a keyboard eltűnését, illetve a begépelt érték hozzárendelését az adott változóhoz! <https://www.cometchat.com/tutorials/how-to-dismiss-ios-keyboard-swift>

**Március 23.**

* Belefutottam egy olyan hibába, hogy a Vstack, Hstack, stb konténerek csak 10 db View elemet (vagy abból származtatott komponenst) tudnak eltárolni -> megoldás: azoknak az elemeknek, amelyek már nem férnének bele az adott konténerbe, létrehozok egy „Group” objektumot, majd ezt az objektumot rakom a konténer utolsó szabad helyére. Azokat a View objektumokat, amelyeket hozzá szeretnék adni a konténerhez, egyszerűen csak belerakom a Group objektumba, és a kapott hiba megszűnik.
* Hozzáadtam a képernyőhöz egy „List” objektumot, amelyben az egyéb kívánt keresési paramétereket lehet majd kiválasztani (nem-dohányzó, állatbarát hely, stb).

<https://developer.apple.com/documentation/swiftui/list>

* + A „List” objektummal felmerült egy olyan probléma, hogy az egész View-ban alkalmazott ScrollView és a List olyan módon nem voltak kompatibilisek egymással, hogy a ScrollView a dimenzióit a benne található komponensektől várja, míg a List az őt befoglaló konténertől. Emiatt a List-nek nem volt definiálva méret, ami miatt nem lett renderelve a UI-ra. Erre megoldás, hogy definiáltam egy környezeti változót, amelyben megadtam a List egy sorának minimum magasságát, amelyet egy „frame” modifier-ben átadtam a List-nek.

<https://stackoverflow.com/questions/61437905/swiftui-list-is-not-showing-any-items>

* + További probléma, volt hogy a „NavigationView”-t, amelyet a „List”-hez használtam, nem ismerte fel a fordító, így teljesen ki kellett vinni a ScrollView-n kívülre.
  + A „List” elemeinek kiválasztása az „Edit” gomb megnyomásával történik. A kiválasztás egy „@State” változóban lesz eltárolva, ezáltal megvalósítva a többszörös kiválasztást, azonban a „Done” gomb megnyomásával eltűnik a kiválasztás a UI-n, de a változóban megőrzödik, újbóli „Edit” gomb megnyomással ismét megjelenik.
* Ezen felül hozzáadtam a Foglalás képernyőhöz egy gombot is, amellyel majd a keresést végre lehet hajtani. A gomb design-ja egyelőre meglehetősen kezdetleges, ezen majd javítok, hogy legyen kerete, háttérszín és a betűk színe is megfelelő legyen.

Feladat: „EgyebIgeny” osztály kiszervezése külön Swift fájlba, az egyes tab-oknak megfelelő view-k kiszervezése külön View fájlokba

* Segítség: <https://www.hackingwithswift.com/quick-start/swiftui/adding-tabview-and-tabitem>

**Március 30.**

* Átneveztem a magyar elnevezésű változókat, privát struct-okat az egyes View struct-okban angolra.
* Létrehoztam a szoba kiadási oldalt a tabView-ban:
  + Létrehoztam egy NavigationView objektumot és ScrollView objektumot, hogy a görgetést és az oldalon belüli navigáció megvalósítható legyen
  + Létrehoztam egy „Group” objektumot a földrajzi adatoknak, ahol meg lehet adni különböző TextField objektumokban a kiadni kívánt szállás lokációjában a várost, az irányítószámot, az utca nevet, illetve a földrajzi szélességet és hosszúságot.
  + Létrehoztam egy „Picker” objektumot, amelyben a felhasználó kiválaszthatja a kiadni kívánt szállás típusát, amely jelen pillanatban a következők lehetnek: „Apartman”, „Szoba”, „Kanapé”.
  + Az elfogadott fizetési módokat megint kiszerveztem egy „Group” objektumba, amelyben egy Text objektumban megadtam a nevét ennek a szekciónak a képernyőn, majd egy „List” objektumban kiválaszthatja a felhasználó az elfogadni kívánt fizetési módokat. A fizetési módokhoz létrehoztam egy külön privát struct-ot „AcceptedPaymentMethods” néven, amelyben eltárolom az adott objektum nevét és egy hozzárendelt azonosítót. Ezután egy „@State” tömbben létrehoztam AcceptedPaymentMethods objektumokat, illetve egy másik „@State” változóban (ami egy id-ket eltároló Set) eltárolom a már kiválasztott fizetési módok id-jait. Ez a lista jelen pillanatban még mindig az „Edit” gombos kiválasztási módszerrel működik, amely a későbbiekben további fejlesztésre szorul.
  + Létrehoztam egy TextField objektumot, amelyben meg lehet adni a szállás éjszakánkénti árát.
  + Létrehoztam még egy „Group” objektumot, amelyben a szállás további tulajdonságait lehet felsorolni. Ez a szakasz megegyzik a foglalási oldalon lévő „egyéb igények” résszel. A megvalósítás is hasonló, „List” objektumban lehet kiválasztani a tulajdonságokat, amelyben az objektumok megegyeznek a foglalási oldalban is használt „OtherNeeds” objektummal.
  + Végül pedig elhelyeztem egy gombot az oldal alján, amelyet megnyomva pedig létre lehet hozni a szálláshelyet tartalmazó hirdetést.
  + Hiányzik még: kép rész a szállásról.
* A „ContentView” objektumot „kitakarítottam”: azaz az egyes tab-oknak megfelelő View objektumokat kiszerveztem különálló View fájlokba.
* **Szükséges**: foglalási oldalon az elfogadott fizetési módokat a „Picker” helyett egy „List” objektumba tenni, ezáltal lehetőve tenni, hogy több fizetési módot is „támogasson” a foglaló. Illetve a kiadási oldalon a fizetési módok közé felvenni az „Ellenszolgáltatás” opciót. Date picker-eket és Date adatok formátumát átállítani megfelelő formátumra.

**Április 6.**

* „Profil” tab létre hozása. A „Profil” tabot is a többi tab-nak megfelelően kezdtem el létrehozni. Először létrehoztam egy NavigationView komponenst, ez van a hierarchia legtetején. Majd a NavigationView komponensbe tettem egy ScrollView komponenst. (Ez a kettő komponens lehet, hogy felesleges, idővel lehet, hogy eltávolításra kerülnek.)
* A tab-on különféle Stack komponenseket helyeztem el. Felülre elhelyeztem egy Hstack-et, ennek a komponensnek a bal oldali pozíciójában találhatóak a felhasználónak az adatai, jobb oldali pozíciójában pedig a felhasználó profil képe (default esetben ez egy system image lesz). A felhasználó adatai külön kisebb Vstack-ekben vannak elrendezve.
* A tabon ezen kívül helyet kapott 3 gomb is, amelyek egyelőre még nem rendelkeznek funkcióval. Ezekkel a gombokkal a következő funkcionalitások lesznek majd elérhetőek: „Adatok módosítása”, „Jelszó módosítása” és „Kijelentkezés”. A gombok kinézetét a későbbiekben esztétikusabbra kell majd alakítani, ehhez példaként szolgálhat az egyik megoldás a következő stack overflow-n közzétett problémára: <https://stackoverflow.com/questions/65135725/how-to-position-my-button-to-the-bottom-of-the-screen-swiftui>

**Április 20.**

* Egy új, esztétikusabb „Foglalás” tab létrehozása.
  + Ennél a tab-nál a kiindulási View egy Form volt, ezen a Form-on belül hoztam létre Section-öket. Az első Section-ben lehet megadni a várost, ahol a szállást keresi a felhasználó, amelyet egy TextField-ben lehet megadni. Ugyanebben a Section-ben van lehetősége a felhasználónak kiválasztani a látogatás kezdetét és végét, erre 1-1 DatePicker objektum használatával van lehetőség. Majd meg kell adni a fizetés módját, ez a korábbiakkal analóg módon történik, egy Picker segítségével.
  + A következő Section-ben lehetőség van megadni preferált árkategóriát is. Mivel ezt nem kötelező megtenni, és mivel előfordulhat olyan eset, hogy csak alsó, vagy csak felső határt szeretne megadni a felhasználó, ezért 1-1 Toggle-t helyeztem el a UI-n, amelyek „On” helyzetbe állításával megjelenik a TextField, ahol a felhasználó megadhatja a határnak szánt értéket. Amennyiben nem kíván ilyen határértékeket megadni a felhasználó, úgy ezeket a Toggle-ket „Off” helyzetben hagyva nem is jelenik meg a számár a TextField, ahol meg lehet adni az értékeket.
  + A preferált értékelés esetében a preferált árkategória megadásánál alkalmazott módszerrel analóg módon jártam el.
  + Az „Egyéb igények” részt is egy külön Section-ben helyeztem el. Ebben a Section-ben létrehoztam egy List View-t amelyben a felmerülő egyéb igényeket soroltam fel. Ezt oly módon tettem, hogy az egyéb igényeket felsoroló tömbön végig iteráltam egy ForEach ciklussal, majd minden iterációban az adott elemre létrehozok egy gombot, amely megnyomása esetén az adott elem kiválasztottsági státuszát módosítja (ha előtte nem volt kiválasztva, akkor ki lesz választva, és fordítva), illetve a gomb megjelenéséhez egy újonnan létrehozott View struct-ot alkalmaztam. Az új View struct a „OtherNeedsListItemSelectedView” nevet kapta, és ennek az objektumtípus arra szolgál, hogy egy egységes kinézetet valósítson meg az OtherNeed típusú objektumoknak az ilyen kiválasztásos listás esetekben. Amennyiben egy ilyen objektum kiválasztásra kerül, akkor egy narancssárga pipa kerül a sor elejére, amelyet az objektum neve követ. Ha nincs kiválasztva az objektum, akkor csak az objektum neve jelenik meg a sor elején.

Ennek az igényesebb kiválasztásos módszernek a megvalósítása nem kevés gondolkodást igényelt, lévén, hogy natívan nincs olyan módszer, amely támogatna efféle kiválasztásos struktúrát, illetve a korábbi „Edit” gombbal való kiválasztáshoz képest is jelentősen felhasználóbarátabb.

* + Végül hozzáadtam egy „Keresés” gombot ehhez a képernyőhöz is, amely megnyomásával majd kilistázódnak a keresés paramétereinek megfelelő szálláshelyek.
  + Ezen felül meg lett oldva, hogy a felhasználó csak számokat tudjon begépelni olyan TextField-ekbe, ahol csak numerikus inputnak van értelme. Illetve a billentyűzethez hozzá lett adva toolbar-ban egy „Done”, amely megnyomásával megvalósul az értékadás a TextField-en keresztül a hozzá kapcsolt @State változónak.
* Létrehoztam az előzővel közel analóg módon egy „Szoba kiadása” View-t is.
  + Itt is a Form-os megközelítést alkalmaztam, csakúgy, mint az előző esetben. Itt az első Section-ben a szállás földrajzi adatait várja az alkalmazás a felhasználótól. 5 darab TextField-ben kell megadnia a várost, az irányítószámot, az utcát, és a földrajzi koordinátákat.
  + A következő Section-ben egy Picker segítségével a felhasználó kiválaszthatja a meghirdetni kívánt szállás típusát: apartman, szoba vagy kanapé.
  + A harmadik Section-ben ki lehet választani azokat a fizetési módokat, amelyeket a szoba kiadója elfogad. Itt is analóg megoldást alkalmaztam, mint a foglalási View esetében, amikor az egyéb igényeket kellett megadni. Egy List View-ban kilistázom a lehetőségeket, amelyet egy ForEach ciklus segítségével valósítok meg. A List-ben gombokat helyezek el, amely az adott fizetési mód kiválasztottsági státuszát módosítja, és a gomb megjelenése tulajdonképpen egy Hstack, amelynek elején kiválasztás esetén egy narancssárga pipa jelenik meg, majd a fizetési mód neve, kiválasztatlan állapotban pedig a pipa nem jelenik meg.
  + A negyedik Section-ben meg lehet adni az éjszakánkénti árat, az ötödik Section-ben pedig a szoba tulajdonságait lehet kiválasztani, amelyek lefedik a foglalási oldalon szereplő „egyéb igényeket”. A megvalósítás is analóg módon történik a foglalási oldal „egyéb igényeket” kilistázó Section-jével.
  + Végül pedig az utolsó Section-ben elhelyeztem egy gombot, amely megnyomásával majd meghirdetésre kerülhet a szállás.
  + A numerikus bevitel megvalósítása ebben az esetben is analóg módon történik a foglalási oldalon való megvalósítással.

**Április 27.**

* Custom színek hozzáadása az asset-ekhez, osztály létrehozása, amelyen keresztül elérhetőek a különböző View objektumok modifier-eiből. Ezeket a színeket felhasználva esztétikusabb megjelenést alakítottam ki az alkalmazásban megjelenő gomboknak, köztük a Form objektumokon belül szereplő gomboknak is.
* Ranglista nézet („LeaderboardView”) kialakítása: az oldal tetején egy Picker található „.segmented” stílusban, ezzel választható ki, hogy a bérlők/foglalók vagy a kiadók ranglistájára kíváncsi a felhasználó. Amikor kiválasztotta, akkor egyelőre egy példa usereket tartalmazó tömb alapján kilistázza az alkalmazás a userek adatait képpel, névvel és az adott ranglistához tartozó pontszámmal.
* A userek ranglista adataihoz külön View objektumot („UserLeaderboardView”) hoztam létre. Ez megjeleníti a user profilképét, amely default esetben egy system kép, a felhasználó nevét és pontszámát. A View egy Hstack-ből áll, amelyben baloldalon található a profilkép, és jobboldalon egy Vstack, amelyben egymás alatt találhatóak a nevet és a pontszámot tartalmazó Text objektumok.
  + A userek adatait egy külön erre a célra elkészített objektum alapján szerzi meg a ranglistán szereplő user adatokat megjelenítő View. Ebben szerepelnek a korábban már felsorolt adatok, illetve egy UUID, amely alapján azonosítani lehet az objektumokat (és így pl. ForEach objektumban lehet őket használni).

Elvégzendő feladat: a példa usereket tartalmazó tömböket kiszervezni egy külön objektumba, akár egy environment változóba, és a pontszám alapján történő sorbarendezést elvégezni rajtuk. Erre a View megjelenítésekor, illetve módosításakor nincs lehetőség, így külön objektum kell majd erre a célra. Illetve, a későbbiekben, amikor nem példa tömbökkel fog működni az alkalmazás, akkor is könnyebb külön karbantartani az erre a célra létrehozott objektumot és adott esetben frissíteni azt. (Tervezési minták alkalmazása.)