INFOtec Informatikai és Információtechnikai Kft

Identity Hungary Kft. - GINOP-2.1.7-15-2016-02650 GarNET – egységes garanciakezelő kártya rendszer

FELADAT 2

MOBIL ALKALMAZÁS KIALAKÍTÁSA 1,3 részfeladat

Fejlesztői dokumentáció

- **Login**

Bejelentkezés

> Az oldal lehetővé teszi a már meglévő felhasználóval való bejelentkezést az alkalmazásba.

Elemek

- GarNet logó
- E-mail mező
- Jelszó mező
- Emlékezz rám kapcsoló
- Elfelejtettem a jelszavam link
- Bejelentkezés gomb
- Navigációs sáv

Metódusok

loginHandler

Összegyűjti a formba beírt adatokat, és meghívja a login végpontot.

> ✅ **Sikeres hívás:** a felhasználót átirányítjuk a dashboard oldalra.
> ❌ **Sikertelen hívás:** megjelenítjük a backendről érkező hibaüzenetet.

Regisztráció

> Az oldal lehetővé teszi új GarNet felhasználók regisztrációját.

Elemek

- GarNet logó
- E-mail mező
- Jelszó mező
- Jelszó újra mező
- Regisztráció gomb
- Navigációs sáv

Metódusok

regHandler

Összegyűjti a formba beírt adatokat, és meghívja a regisztráció végpontot.

> ✅ **Sikeres hívás:** a felhasználót átirányítjuk a bejelentkezés oldalra.
> ❌ **Sikertelen hívás:** megjelenítjük a backendről érkező hibaüzenetet.

```
    remember_me: loginForm.value.remember_me
  };

  userService.login(body).then(success => {
    if (success) {
      storage.set('login-data', {
        remember_me: body.remember_me,
        email: body.email,
        password: body.password
      });
      navCtrl.navigateRoot('dashboard',
        {
          animated: true,
          animationDirection: 'forward'
        }
      );
    }
  });
}
```

```typescript
regHandler(): void {
  const body = {
    email: regForm.value.email,
    password: regForm.value.password,
    password_confirmation: regForm.value.password,
    role_id: '2'
  };

  userService.register(body).then(async success => {
    if (success) {
      segmentChanged('0');
    }
  });
}
```

Elfelejtett jelszó

> Az oldal lehetővé teszi új jelszó igénylését meglévő felhasználóhoz.

Elemek

- GarNet logó
- E-mail mező
- Mehet gomb
- Navigációs sáv

Metódusok

forgotHandler

Összegyűjti a formba beírt adatokat, és meghívja az elfelejtett jelszó végpontot.

> ✅ **Sikeres hívás:** a felhasználót átirányítjuk a bejelentkezés oldalra.
> ❌ **Sikertelen hívás:** megjelenítjük a backendről érkező hibaüzenetet.

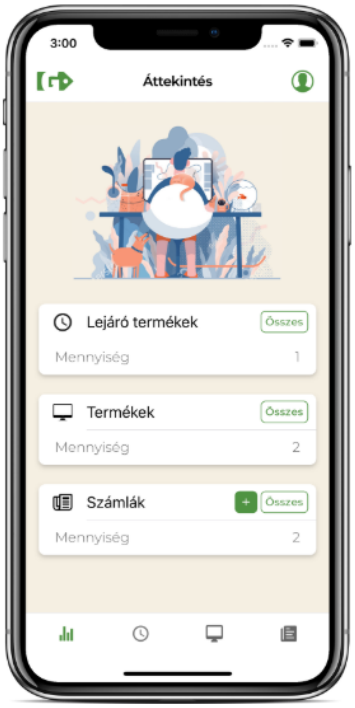
```
  };

  userService.forgot(body).then(success => {
    if (success) {
      segmentChanged('0');
      forgotForm.patchValue({
        email: ''
      });
    }
  });
}
```

- **Dashboard**

Áttekintés

> Az oldalon megtekinthetjük a felhasználóhoz tartozó összesítést,
> vehetünk fel új számlát, és megnyithatjuk a profilunkat megtekintésre.



Elemek

- **Fejléc**
  - Profil gomb
- Számlák kártya
- Termékek kártya
- Lejáró termékek kártya
- Navigációs sáv

Metódusok

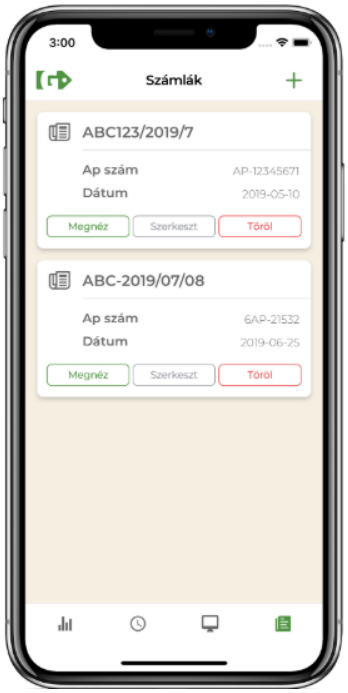
getCounters

> Feliratkozik a számlálók változását követő eseményre.

```typescript
getCounters(): void {
  this.events.subscribe('counterChange', counter => {
    this.count[counter.list] = counter.count;
  });
}
```

Számlák

> Az oldalon megtekinthetjük a felhasználóhoz tartozó számlák listáját, és vehetünk fel újat.



Elemek

- **Fejléc**
  - Új számla gomb
- Számlák listája
- Navigációs sáv

Metódusok

refreshList

> Lekéri a felhasználóhoz tartozó számlák listáját

```typescript
async refreshList(): void {
  if (!user) {
    const user = await storage.get('user');
    user = user;
  }

  invoicesService.getList(user.user_id, 1)
    .then((res: ListDataResponse) => {
    events.publish('counterChange',
      {
        list: 'invoices',
        count: res.total
      });

    listData = res.data;
    loading = false;
  });
}
```

openInvoice

> **Megnyitja a kiválasztott számlát megtekintésre vagy szerkesztésre.**

> - **editMode**: A megnyomott gombtól (Megnéz, Szerkeszt) függően igaz vagy hamis
> - **idx**: A kiválasztott számla indexe

typescript

```
async openInvoice(editMode, idx): Promise<any> {
  const invoiceItems = await invoicesService.getInvoiceItems(
    user.user_id,
    listData[idx].id
  );
  const modal = await modalCtrl.create({
    component: InvoiceModalComponent,
    componentProps: {
      editMode,
      invoice: listData[idx],
      invoiceItems,
      sellers: sellers,
      user: user
    }
  });
  modal.present();
}
```

showRemoveAlert

> Megnyitja a törlés megerősítése felugró ablakot.

> - **idx**: A kiválasztott számla indexe

```typescript
showRemoveAlert(idx): void {
  modalService.openModal('delete').then(success => {
    if (success) {
      deleteInvoice(idx)
    }
  });
}
```

deleteInvoice

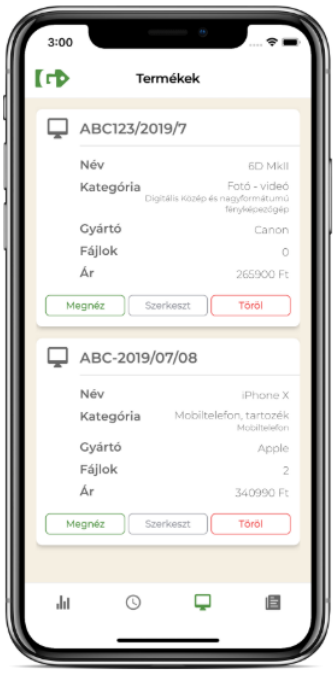
> Meghívja a számla törlése végpontot, majd frissíti a nézetet.

> - **idx**: A kiválasztott számla indexe

```typescript
deleteInvoice(idx): void {
  const body = {
    user_id: user.user_id,
    invoice_id: listData[idx].id
  };
  invoicesService.deleteInvoice(body).then(() => {
    refreshList();
    events.publish('updateHeight');
  });
}
```

Termékek

> Az oldalon megtekinthetjük a felhasználóhoz tartozó számlák listáját, és vehetünk fel újat.



Elemek

- Fejléc
- Termékek listája
- Navigációs sáv

Metódusok

refreshList

> Lekéri a felhasználóhoz tartozó termékek listáját

```typescript
async refreshList(): void {
  if (!user) {
    const user = await storage.get('user');
    user = user;
  }

  productsService.getList(user.user_id, 1)
    .then((res: ListDataResponse) => {
      events.publish('counterChange',
      {
        list: 'products',
        count: res.total
      });
    listData = res.data;
    loading = false;
  });
}
```

openProduct

> Megnyitja a kiválasztott terméket megtekintésre vagy szerkesztésre.

> - **editMode**: A megnyomott gombtól (Megnéz, Szerkeszt) függően igaz vagy hamis
> - **idx**: A kiválasztott termék indexe

typescript

```typescript
async openProduct(idx, editMode): void {
  const selected = listData[idx];
  const categories = await productsService.getProductCategories();
  const extraFormItems =
    await productsService.getExtraFormItems(selected.product_subcategory_id);
  const modal = await modalCtrl.create({
    component: ProductModalComponent,
    componentProps: {
      canEdit: true,
      categories,
      editMode,
      extraFormItems,
      product: selected,
      user: user
    }
  });
  modal.present();
}
```

showRemoveAlert

> Megnyitja a törlés megerősítése felugró ablakot.

> - **idx**: A kiválasztott termék indexe

```typescript
showRemoveAlert(idx): void {
  modalService.openModal('delete').then(success => {
    if (success) {
      deleteInvoice(idx)
    }
  });
}
```

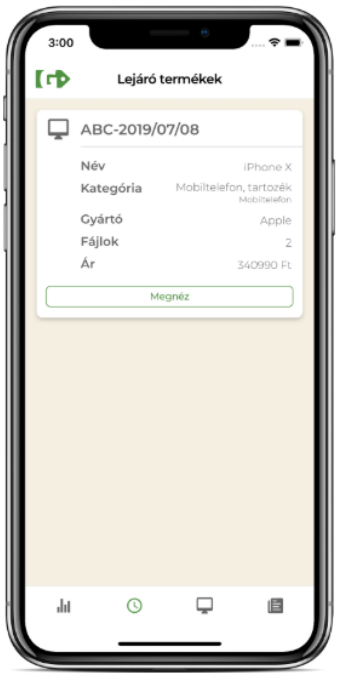
deleteProduct

> Meghívja a számla törlése végpontot, majd frissíti a nézetet.

> - **idx**: A kiválasztott termék indexe

```typescript
deleteProduct(idx): void {
  const selected = listData[idx];
  const body = {
    user_id: user.user_id,
    invoice_id: selected.invoice_id,
    item_id,
    selected.id
  };
  productsService.deleteProduct(body).then(() => {
    refreshList();
    events.publish('updateHeight');
  });
}
```

Lejáró termékek



> Az oldalon megtekinthetjük a felhasználóhoz tartozó termékeket, amik a beállított
> lejárati értesítés idején belül esik

Elemek

- Fejléc
- Lejáró termékek listája

Metódusok

refreshList

> Lekéri a felhasználóhoz tartozó lejáró termékek listáját

```typescript
async refreshList() {
    if (!user) {
      const user = await storage.get('user');
      user = user;
    }

    expiringProductsService.getList(user.user_id, 1)
      .then((res: ListDataResponse) => {
        events.publish('counterChange',
          {
            list: 'expiring',
            count: res.total
          });
        listData = res.data;
        loading = false
    });
  }
```

openProduct

> Megnyitja a lejáró terméket megtekintésre.

> - **idx**: A kiválasztott termék indexe

typescript

```
async openProduct(idx) {
  const selected = listData[idx];
  const categories = await productsService.getProductCategories();
  const extraFormItems =
    await productsService.getExtraFormItems(selected.item.product_category_id);
  const modal = await modalCtrl.create({
    component: ProductModalComponent,
    componentProps: {
      canEdit: false,
      categories,
      editMode: false,
      extraFormItems,
      product: selected.item,
      user: user
    }
  });
  modal.present();
}
```

- **Komponensek**

File Upload

> A komponens segítségével képeket és PDF dokumentumokat tölthetünk fel.

Elemek

- Hozzáadás gomb
- **Feltöltésre kiválasztott fájlok listája**
  - Kiválasztott kép előnézete / PDF ikon
  - Kiválasztott fájl neve
  - Kiválasztott fájl elvetése gomb

Metódusok

fileChanged

> Új fájl hozzáadásakor generál egy listát a kiválasztott fájlokból.

> - **idx**: Szabványos file change esemény

```typescript
fileChanged(event) {
  const files = event.target.files;
  Array.from(files).forEach((file: any) => {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onloadend = () => {
      files.push({
        thumbnail: reader.result,
        name: file.name
      });
    };
  });

  Array.from(files).forEach((file: any) => {
    formFiles.push(file);
  });
  events.publish('updateHeight');
}
```

showRemoveAlert

> Megnyitja a törlés megerősítése felugró ablakot.

> - **idx**: A kiválasztott csatolmány indexe

```typescript
showRemoveAlert(idx) {
  modalService.openModal('delete').then(res => (res ? removeSelected(idx) : null));
}
```

removeSelected

> Eltávolítja a kiválasztott csatolmányt a listából.

> - **idx**: A kiválasztott csatolmány indexe

```typescript
removeSelected(idx) {
  if (idx > -1) {
    files.splice(idx, 1);
    formFiles.splice(idx, 1);
  }
}
```

Header

> A fejléc komponens az egész alkalmazásban jelen van (leszámítva a belejentkezést).
> A rajta található akciógombok alkalmazkodnak a kontextushoz.

Elemek

- Bal oldali gomb (vagy logó)
- Jobb oldali gomb

Metódusok

buttonClicked

> Elküld egy esemény értesítőt, az adott gomb pozíciójával

> - **side**: A megnyomott gomb pozíciója

```typescript
buttonClicked(side: string): void {
  if (buttonHandler) {
    buttonHandler.emit(side === 'left');
  }
}
```

Image Modal

> A komponens lehetővé teszi a letöltött képek megjelenítését.

Elemek

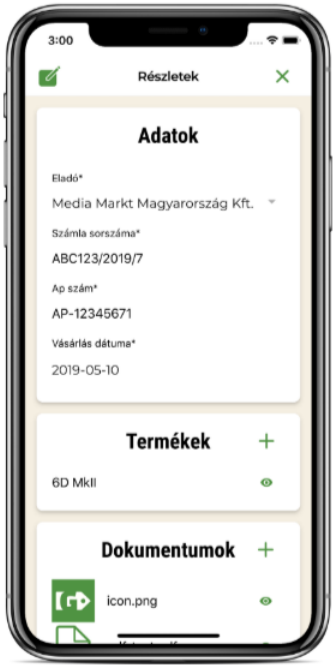
- **Fejléc**
  - Modal bezárása gomb
- Kép

Invoice Modal

> Számla modal komponens

Elemek

- **Fejléc**
  - Szerkesztés / Elküldés gomb
  - Bezárás gomb
- **Adatok kártya**
- **Termékek kártya**
  - Új termék gomb
  - Termékek listája
- **Dokumentumok kártya**
  - Új dokumentum gomb
  - Feltöltött dokumentumok listája



Metódusok

ngOnInit

> Inicializálja a számla modal ablakot, beállítja megjelenítendő értékeket, feliratkozik a nézet frissítéséhez szükséges eseményekre.

typescript

```
ngOnInit(): void {
    translate.stream('invoice.header').subscribe(res => {
      headerTitles = res;
    });

    const invoice = invoice;
    invoiceEditForm = formBuilder.group({
      ap_no: invoice.ap_no,
      invoice_id: invoice.id,
      invoice_no: invoice.invoice_no,
      purchased: invoice.purchased,
      seller_id: invoice.seller_id,
      user_id: user.user_id
    });

    events.subscribe('refreshInvoice', () => {
      invoiceService
        .getInvoiceItems(user.user_id, invoice.id)
        .then(res => (invoiceItems = res));
      invoiceService.getInvoice(user.user_id, invoice.id).then(res => (invoice = res));
    });

    events.subscribe('addAttachment', () => {
      editMode = true;
      setTimeout(() => content.scrollToBottom(100), 100);
    });

    buttonHandler.subscribe(side => {
      buttonClicked(side);
    });
  }
```

editHandler

> Lekezeli a szerkesztés műveletetét, eldönti, hogy milyen változtatásokat kell elküldeni a backend számára.

typescript

```typescript
editHandler(): void {
  const form = invoiceEditForm;
  const formTouched = form.touched;

  if (!formTouched && fileUploader.files.length === 0) {
    editMode = false;
    return;
  }

  if (formTouched) {
    modifyInvoice();
  }
  if (fileUploader.files.length > 0) {
    modifyAttachments();
  }
}
```

newProduct

> Megnyitja az új termék felvétele modal ablakot.

typescript

```typescript
async newProduct(): void {
  const productCategories = await productsService.getProductCategories();
  const modal = await modalCtrl.create({
    component: NewProductComponent,
    componentProps: {
      user: user,
      invoiceId: invoice.id,
      categories: productCategories
    }
  });
  modal.present();
}
```

openProduct

Megnyitja a kiválaszott terméket megtekintésre.

idx A kiválasztott termék indexe

typescript

```
async openProduct(idx): void {
  const selected = invoiceItems[idx];
  const modal = await modalCtrl.create({
    component: ProductModalComponent,
    componentProps: {
      user: user,
      product: selected,
      editMode: false,
      canEdit: true
    }
  });
  modal.present();
  }
```

modifyInvoice

> A számla alap adatainak módosítását kezeli.
> ✔️ **Sikeres hívás**: frissítjük a nézetet.
> ❌ **Sikertelen hívás**: megjelenítjük a backendről érkező hibaüzenetet a hibás mezőknél.

typescript

```
modifyInvoice(): void {
  const form = invoiceEditForm;

  form.patchValue({
    purchased: moment(form.value.purchased).format('YYYY-MM-DD')
  });

  formErrors = {};
  invoiceService
    .modifyInvoice(form.value)
    .then(() => {
      editMode = false;
      events.publish('refreshInvoices');
    })
    .catch(err => (formErrors = err.error.errors || {}));
}
```

modifyAttachments

> A számlához tartozó csatolmányok feltöltéséért felel.
> ✔️ **Sikeres hívás:** frissítjük a nézetet az új dokumentumokkal.
> ❌ **Sikertelen hívás:** megjelenítjük a backendről érkező hibaüzenetet.

typescript

```
modifyAttachments(): void {
  const formData = new FormData();
  formData.append('invoice_id', invoice.id.toString());
  formData.append('user_id', user.user_id.toString());
  fileUploader.formFiles.forEach((file: any) => {
    formData.append('files[]', file, file.name);
  });

  invoiceService.addAttachment(formData, user.user_id).then(() => {
    editMode = false;
    fileUploader.formFiles = [];
    fileUploader.files = [];
    events.publish('refreshInvoices');
    events.publish('refreshInvoice');
  });
}
```

showRemoveAlert

> Megnyitja a törlés megerősítése felugró ablakot.

> - **idx**: A kiválasztott termék indexe

typescript

```
showRemoveAlert(idx) {
  modalService.openModal('delete')
    .then(success => {
      if (success) {
        deleteProduct(idx);
      }
    });
}
```

deleteProduct

> Eltávolítja a kiválasztott terméket.
> ✔️ **Sikeres hívás:** frissítjük a nézetet.
> ❌ **Sikertelen hívás:** megjelenítjük a backendről érkező hibaüzenetet.

typescript

```
async deleteProduct(idx): void {
    const product = this.invoiceItems[idx];
    const body = {
      invoice_id: product.invoice_id,
      item_id: product.id,
      user_id: this.user.user_id
    };

    this.productsService.deleteProduct(body).then(() => {
      this.invoiceItems.splice(idx, 1);
      this.events.publish('refreshProducts');
    });
  }
```

buttonClicked

> A fejlécben található gombok megnyomásának kontextus szerinti lekezelése.

> **isLeftSide:** megadja a megnyomott fejléc gomb pozícióját

typescript

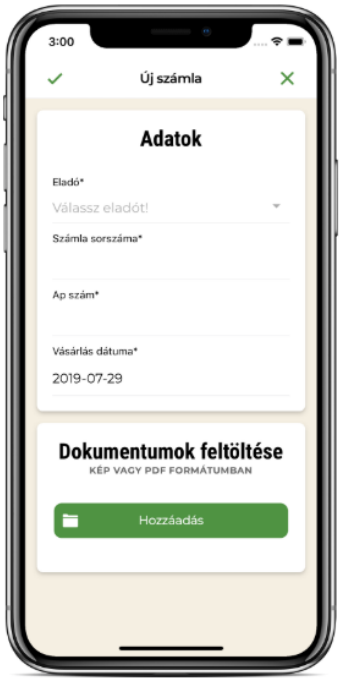
```
buttonClicked(isLeftSide): void {
  if (!isLeftSide) {
    modalCtrl.dismiss();
    return;
  }
  if (isLeftSide && editMode === false) {
    editMode = true;
    return;
  }
  if (isLeftSide && editMode === true) {
    editHandler();
  }
}
```

New Invoice

> Új számla komponens

Elemek

- Fejléc
  - Elküldés gomb
  - Bezárás gomb
- Adatok kártya
- Dokumentumok feltöltése kártya
  - Új dokumentum gomb
  - Feltöltésre kiválaztott dokumentumok listája



Metódusok

ngOnInit

> Inicializálja az új számla modal ablakban található formot,
> és feliratkozik a fejlécben található gombok megnyomására.

typescript

```
ngOnInit() {
    newInvoiceForm = formBuilder.group({
      ap_no: null,
      files: null,
      invoice_no: null,
      purchased: maxPurchaseDate, // mai dátum
      seller_id: null,
      user_id: user.user_id
    });

    buttonHandler.subscribe(side => {
      buttonClicked(side);
    });
  }
```

sendForm

> Összegyűjti a formba bevitt adatokat, a backend által megkövetelt formátumra alakítja, majd elküldi az adatokat.

> ✅ **Sikeres hívás**: visszanavigáljuk a usert a listanézetre.
> ❌ **Sikertelen hívás**: megjelenítjük a hibás mezők alatt a backendről érkező hibaüzenetet.

typescript

```typescript
sendForm() {
    newInvoiceForm.patchValue({
      files: fileUploader.formFiles,
      purchased: moment(newInvoiceForm.value.purchased).format('YYYY-MM-DD')
    });
    const form = newInvoiceForm.value;

    const formData = new FormData();
    Object.keys(form).forEach(el => {
      if (el === 'files') {
        Array.from(form[el]).forEach((file: any) => {
          formData.append('files[]', file, file.name);
        });
      } else {
        formData.append(el, form[el]);
      }
    });

    invoiceService
      .addInvoice(user.user_id, formData)
      .then(() => {
        events.publish('refreshInvoices');
        events.publish('updateHeight');
        modalCtrl.dismiss();
      })
      .catch(err => {
        formErrors = err.error.errors || {};
      });
  }
```

buttonClicked

> A fejlécben található gombok megnyomásának kontextus szerinti lekezelése.

> **isLeftSide**: megadja a megnyomott fejléc gomb pozícióját

typescript

```
buttonClicked(isLeftSide) {
  if (isLeftSide) {
    sendForm();
  }
  if (!isLeftSide) {
    modalCtrl.dismiss();
  }
}
```