IISSI 2 LAB EXAM APPROACH STRATEGY

BACKEND

MODEL

```
Product.init({
    name: DataTypes.STRING,
    grasas: {
      allowNull: false,
      type: DataTypes.DOUBLE,
      defaultValue: 0.0
    },
    proteinas: {
      allowNull: false,
      type: DataTypes.DOUBLE,
      defaultValue: 0.0
    },
    carbohidratos: {
      allowNull: false,
      type: DataTypes.DOUBLE,
      defaultValue: 0.0
    },
    calorias: {
      allowNull: false,
      type: DataTypes.DOUBLE,
      defaultValue: 0.0
    },
```

MIGRATION

```
await queryInterface.createTable('Products', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      name: {
        allowNull: false,
        type: Sequelize.STRING
      },
      grasas: {
        type: Sequelize.DOUBLE,
        defaultValue: 0.0
      },
      proteinas: {
        type: Sequelize.DOUBLE,
        defaultValue: 0.0
```

```
          type: Sequelize.DOUBLE,
          defaultValue: 0.0
        },
        calorias: {
          type: Sequelize.DOUBLE,
          defaultValue: 0.0
        },
        description: {
          type: Sequelize.TEXT
        },
```

SEEDERS

```
{ name: 'Ensaladilla', grasas: 10, proteinas: 10, carbohidratos: 10,
calorias: 10, description: 'Tuna salad with mayonnaise', price: 2.5,
image: process.env.PRODUCTS_FOLDER + '/ensaladilla.jpeg', order: 1,
availability: true, restaurantId: 1, productCategoryId: 1 },
        { name: 'Olives', description: 'Home made', price: 1.5, image:
process.env.PRODUCTS_FOLDER + '/aceitunas.jpeg', order: 2, availability:
true, restaurantId: 1, productCategoryId: 1 },
```

VALIDATOR

```
const checkCaloriasInBound = async (req) => {
  try {
    const grasas = req.body.grasas
    const carbohidratos = req.body.carbohidratos
    const proteinas = req.body.proteinas
    if (9 * grasas + 4 * carbohidratos + 4 * proteinas > 1000) {
      return Promise.reject(new Error('Danger!! Potential of
overweight'))
    } else { return Promise.resolve() }
  } catch (err) {
    return Promise.reject(new Error('Error while checking calorias'))
  }
}

module.exports = {
  create: [
    check('name').exists().isString().isLength({ min: 1, max: 255
}).trim(),
    check('grasas').isFloat().toFloat(),
    check('proteinas').isFloat().toFloat(),
    check('carbohidratos').isFloat().toFloat(),
    check('calorias').custom((value, { req }) => {
```

```
update: [
    check('name').exists().isString().isLength({ min: 1, max: 255 }),
    check('proteinas').isFloat().toFloat(),
    check('carbohidratos').isFloat().toFloat(),
    check('calorias').custom((value, { req }) => {
      return checkCaloriasInBound(req)
    }),
    check('description').optional({ nullable: true, checkFalsy: true
}).isString().isLength({ min: 1 }).trim(),
    check('price').exists().isFloat({ min: 0 }).toFloat(),
```

CONTROLLER

```
exports.create = async function (req, res) {
  let newProduct = Product.build(req.body)
  newProduct.calorias = 9 * newProduct.grasas + 4 *
newProduct.carbohidratos + 4 * newProduct.proteinas
  if (typeof req.file !== 'undefined') {
    newProduct.image = req.file.destination + '/' + req.file.filename
  }
  try {
    newProduct = await newProduct.save()
    res.json(newProduct)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

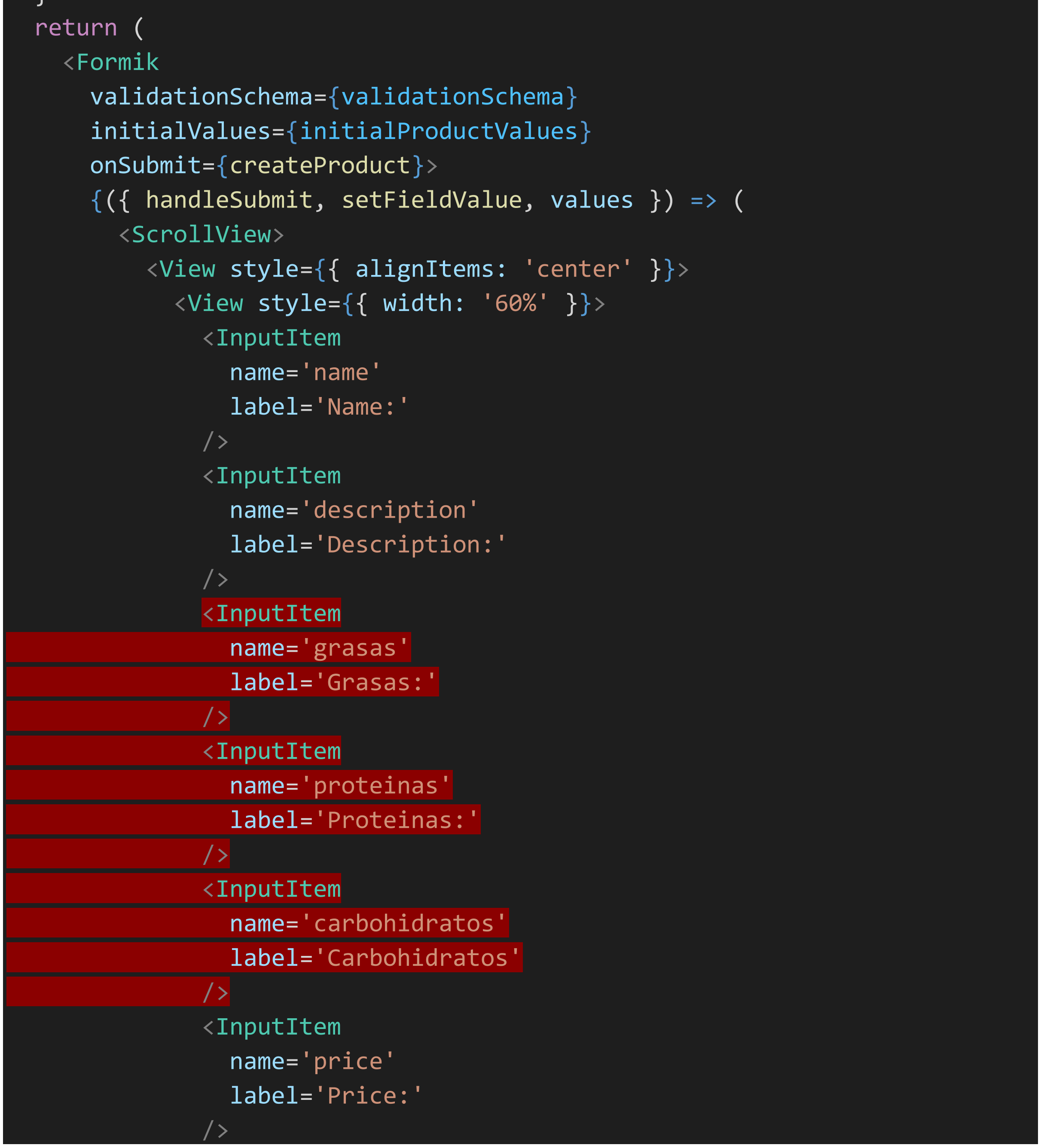
```
exports.update = async function (req, res) {
  if (typeof req.file !== 'undefined') {
    req.body.image = req.file.destination + '/' + req.file.filename
  }
  const grasas = req.body.grasas
  const carbohidratos = req.body.carbohidratos
  const proteinas = req.body.proteinas
  req.body.calorias = 9 * grasas + 4 * carbohidratos + 4 * proteinas
  try {
    await Product.update(req.body, { where: { id: req.params.productId }
})
    const updatedProduct = await Product.findByPk(req.params.productId)
    res.json(updatedProduct)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

RESTAURANT DETAIL SCREEN (CALORIAS)

```
const renderProduct = ({ item }) => {
    return (
      <ImageCard
        imageUri={item.image ? { uri: process.env.API_BASE_URL + '/' +
item.image } : defaultProductImage}
        title={item.name}
      >
        <TextRegular numberOfLines={2}>{item.description}</TextRegular>
        <TextSemiBold
textStyle={styles.price}>{item.price.toFixed(2)}€</TextSemiBold>
        <TextSemiBold textStyle={styles.price}> Grasas: {item.grasas ||
0.0}</TextSemiBold>
        <TextSemiBold textStyle={styles.price}> Proteinas:
{item.proteinas || 0.0}</TextSemiBold>
        <TextSemiBold textStyle={styles.price}> Carbohidratos
{item.carbohidratos || 0.0}</TextSemiBold>
        <TextSemiBold textStyle={styles.price}> Calorias: {item.calorias
|| 0.0}</TextSemiBold>
        {!item.availability &&
          <TextRegular textStyle={styles.availability }>Not
available</TextRegular>
        }
      </ImageCard>
    )
  }
```

CREATE PRODUCT SCREEN (CALORIAS)

```
const createProduct = async (values) => {
    setBackendErrors([])
    try {
      console.log(values)
      const createdProduct = await create(values)
      showMessage({
        message: `Product ${createdProduct.name} succesfully created`,
        type: 'success',
        style: flashStyle,
        titleStyle: flashTextStyle
      })
      navigation.navigate('RestaurantDetailScreen', { id:
route.params.id, dirty: true })
    } catch (error) {
      console.log(error)
```



```
  }
  return (
    <Formik
      validationSchema={validationSchema}
      initialValues={initialProductValues}
      onSubmit={createProduct}>
      {({ handleSubmit, setFieldValue, values }) => (
        <ScrollView>
          <View style={{ alignItems: 'center' }}>
            <View style={{ width: '60%' }}>
              <InputItem
                name='name'
                label='Name:'
              />
              <InputItem
                name='description'
                label='Description:'
              />
              <InputItem
                name='grasas'
                label='Grasas:'
              />
              <InputItem
                name='proteinas'
                label='Proteinas:'
              />
              <InputItem
                name='carbohidratos'
                label='Carbohidratos'
              />
              <InputItem
                name='price'
                label='Price:'
              />
```

guay que para cuando acaben los parciales tu cara esté perfecta . Garnier PureActive